Enhanced performance simulation of diesel engines

Haysom, F. J.

Award date:
1989

Awarding institution:
University of Bath

Enhanced performance simulation of diesel engines.

submitted by F. J. Haysom B.Sc.

for the degree of Ph.D.

at the University of Bath

1989

Francis Haysom

Bath, October 1989.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

SUMMARY.

A new high speed diesel engine simulation for a parallel computer using the 'filling and emptying' method is presented. The new simulation has a distributed structure which allows the accurate representation of the filling and emptying method equations and a high processor utilisation and throughput.

The design of a computer data acquisition system for an instrumented diesel engine is described.

Measurements from the parallel simulation are presented and these are compared with an equivalent serial computer engine simulation and with the results from the instrumented engine.

New methods for increasing the parallelism of the engine simulation are discussed and initial results for these new methods are presented.

ACKNOWLEDGEMENTS.

I would like to thank my supervisor, Mr. A. R. Daniels, and Dr. S. J. Charlton for their help and guidance throughout this work. I should also like to thank my colleagues in both the School of Electrical Engineering and the School of Mechanical Engineering for the interest they have shown in this project.

I wish to thank Professor J. F. Eastham for permission to study in the School of Electrical Engineering and Professor F. Wallace for providing access to the Wolfson engine test bed facilities. The financial support of the Science and Engineering Research Council is gratefully acknowledged.

Finally I wish to thank Fiona Gosling for her help with the proof reading of this thesis and for the encouragement she has given me throughout this research.

CONTENTS

SUMMARY. (ii)

ACKNOWLEDGEMENTS. (iii)

CONTENTS. (iv)

SYMBOLS USED. (ix)

1. INTRODUCTION.

1.1. The microprocessor revolution. 1.2

1.2. Parallel processing. 1.3

1.3. The modern Diesel engine. 1.4

1.4. An outline of the contents of this thesis. 1.6

2. DIESEL ENGINE MODELLING.

2.1. Introduction. 2.2

2.2. The 'filling and emptying' method. 2.3

2.3. The derivation of the generalised 'filling and emptying' control volume equations. 2.5

2.4. The cylinder control volume equations. 2.8

2.5. The manifold control volume equations. 2.12

2.6. The calculation of the terms in the 'filling and emptying' method equations. 2.14

2.7. The dynamic engine model. 2.27

2.8. Advanced Diesel engine modelling techniques. 2.31

3. THE BATH UNIVERSITY PARALLEL COMPUTER (BUPC).

3.1. A description of the computer hardware. 3.2

3.2. A description of the computer software. 3.7

3.3. The limitations of the parallel computer. 3.14

3.4. Future developments. 3.19

4. THE PARALLEL SIMULATION OF A MULTI-VOLUME DIESEL ENGINE.

4.1. Introduction. 4.2

4.2. The integration method used in the parallel diesel engine simulations. 4.4

4.3. Historical development of parallel diesel engine modelling. 4.6

4.4. A critical examination of both the CSSC and CISC parallel diesel engine simulations. 4.11

4.5. The design criteria used in the design of the new diesel engine simulation. 4.17

4.6. A new diesel engine simulation. 4.22

4.7. An analysis of the execution speed of the new DISC simulation. 4.35

4.8. Conclusion. 4.40

5. THE USER INTERFACE TO THE DIESEL ENGINE SIMULATION.

5.1. Introduction. 5.2

5.2. The development of a menu based interface for the parallel diesel engine simulation. 5.2

5.3. An animated graphics display for the diesel engine simulation. 5.4

5.4. The GKS graphics system for display of static display and hard copy of engine simulation results. 5.10

5.5. Diesel engine simulation result organisation. 5.11

5.6. Conclusion. 5.18

6. INCREASED PARALLELISM IN THE SIMULATION OF DIESEL ENGINES.

6.1. Introduction. 6.2

6.2. Increased parallelism in the Diesel engine simulation through control volume equation segmentation. 6.3

6.3. Creating parallelism in the Euler predictor corrector method to achieve maximum speed through optimising integration step length. 6.8

6.4. 'Algorithmic' parallelism in the calculation of ordinary differential equations. 6.9

6.5. The implementation of the block predictor corrector integration method. 6.16

6.6. Changes to the Diesel engine equations needed when using a parallel integration method. 6.26

6.7. The block predictor corrector method applied to the Diesel engine equations. 6.31

6.8. A discussion of how the block predictor corrector method can be applied to a full six cylinder Diesel engine simulation. 6.36

6.9. Conclusions. 6.37

7. THE INSTRUMENTATION AND DATA ACQUISITION SYSTEM FOR THE TL11 TRUCK DIESEL ENGINE.

7.1. Introduction. 7.2

7.2. The Leyland TL11 truck engine. 7.2

7.3. The instrumentation of the Leyland TL11 engine. 7.3

7.4. The engine data acquisition system hardware. 7.13

7.5. The engine data acquisition system software. 7.31

8. THE VALIDATION OF THE PARALLEL ENGINE SIMULATION.

8.1. Introduction. 8.2

8.2. Comparison of the parallel engine simulation with an equivalent serial computer engine simulation. 8.2

8.3. The comparison of the parallel engine simulation with the TL11 engine. 8.6

8.4. Conclusions. 8.10

9. THE APPLICATIONS OF A HIGH SPEED SIMULATION OF A DIESEL ENGINE.

9.1. Introduction. 9.2

9.2. Engine condition monitoring and fault diagnosis in marine engines. 9.2

9.3. A high performance engine simulation as an interactive engine design aid. 9.6

9.4. High performance engine simulation in advanced engine control design. 9.7

9.5. Engine operator training. 9.7

10. CONCLUSIONS.

10.1. Conclusions. 10.2

10.2. Future work. 10.3

REFERENCES

APPENDIX A. Data acquisition card circuit diagrams.

APPENDIX B. A brief specification of the new Bath University shared memory, multi-Transputer computer.

APPENDIX C. The Leyland TL11 Diesel engine specification.

APPENDIX D. The menu utility routines for use in user interfaces.

List of symbols used.

A	Area
BMEP	Brake mean effective pressure.
BSFC	Brake specific fuel consumption.
C	Speed of sound, Constant.
calv	Fuel calorific value.
d	Cylinder bore.
FBR	Fuel burning rate.
FMEP	Friction mean effective pressure.
h	Heat transfer coefficient.
H_o	Stagnation enthalpy
h_o	Specific stagnation enthalpy.
IMEP	Indicated mean effective pressure.
ISFC	Indicated specific fuel consumption.
J	Inertia.
K	Gain.
l	Cylinder connecting rod length.
m	Mass
P	Pressure.
PMEP	Pumping mean effective pressure.
Q	Energy, Heat.
R	Gas constant.
r	Engine crankshaft radius.
T	Temperature.
t	Time.
U	Internal energy.
u	Specific internal energy.

V	Volume.
v	Velocity.
W	Work.
x	Actuator position.
β	Mode of burning factor.
Γ	Compression ratio.
γ	Ratio of specific heats.
η	Efficiency.
λ	Fuel air ratio.
θ	Crankshaft angle
σ	Boltzmann constant.
τ	Torque
τ	Time since combustion start.
ξ	Damping ratio.
ω	Rotational speed.
ω_o	Natural frequency.

List of subscript symbols used.

amb	ambient.
c	compressor.
cexit	compressor exit.
cin	compressor inlet.
cl,cc	cylinder closed period.
com	cylinder combustion period.
crit	critical
cyl	cylinder.
d	downstream.
d	dynamic timing.
e,ev	exhaust valve.
e	engine.
em	exhaust manifold.
evo	exhaust valve open
ex	cylinder exhaust period.
f	fuel.
fb	fuel burnt.
i,iv	inlet valve.
ic	intercooler.
id	ignition delay.
im	inlet manifold.
in	cylinder induction period.
ivc	inlet valve close.
j	junction.
man	manifold.
max	maximum.

mech	mechanical.
oc	open cycle.
pis	cylinder piston.
s	static timing.
sf	surface.
sc	cylinder scavenge period.
t	turbine.
tc	turbocharger.
tin	turbine inlet.
texit	turbine exit.
TDC	top dead centre.
u	upstream.
v	valve throat.
vol	volumetric.
w	wall.

CHAPTER 1.

INTRODUCTION.

1.1. The microprocessor revolution.

The use and application of computers has gone through a revolution since their first appearance after the second world war. They have progressed from being large unreliable machines made of electronic valves to being common place components in a large range of everyday equipment. The development of the microprocessor has probably had the most significant effect on the diversification in the uses for computers. Initially microprocessors were not very powerful with only simple logic and arithmetic sequencing functions. The improvements in fabrication techniques, and the expanding market for microprocessor devices has lowered the price of these components. Simultaneously the computational power of microprocessors has increased. Today's microprocessor incorporates the features of both mini and mainframe computers, with a subsequent blurring in the distinctions between micro, mini and mainframe computers. The Motorola 68030 microprocessor has an equivalent processing power to a medium sized mainframe computer of the early 1970s, on just one chip.

The development in dynamic RAM, the primary storage medium in modern computers, is a good illustration of the recent improvements in computer technology. The packing density of dynamic RAM has roughly doubled in every year since 1975. In 1975, using 1Kbit dynamic memory chips, it would have cost $95,528 for one megabyte of memory and 8,192 chips would have been used. By 1986, with 1024kbit dynamic memory chips, this cost had reduced to $240 with a chip count of only eight [1].

The relative cheap price of the modern microprocessor has allowed computing hardware to be dedicated to just one application. For example, the modern washing machine is controlled by a microprocessor which has replaced the electro-mechanical controller. The use of a microprocessor has given an increase in the reliability of the controller and the variety of washing programmes available. It has also allowed the mechanical parts of the machine to be treated more kindly with the implementation of more controlled startups and the inclusion of basic fault diagnosis.

Microprocessors have completely transformed certain product markets. One notable example of this has been the development of the electronic calculator market. Initially introduced in the early 1960s, electronic calculators were priced in excess of £1,000 each. These early calculators were desktop machines, but their reliability and superior performance, over that of their mechanical predecessors, led to a rapid increase in demand. The expanding market for calculators allowed a large amount of technical development in both manufacturing and in the range of functions provided by the calculator. This has led to a massive decrease in the cost of the calculator. Today's calculator has almost become a give away item, with the price being largely determined by the packaging and retailing costs.

1.2. Parallel processing.

Increasingly, the speed of the fastest computers is being restricted by physical barriers, such as the speed of light. However, the demand for ever increasing computing resources, to solve more difficult problems, shows no signs of stopping as is shown by the expanding market for super-computers. The problem is that the increases in processing power are now being achieved at an exponential

increase in cost. An obvious solution to these physical limitations is the use of more than one processor to solve a problem, a parallel computer. Two processors should theoretically do the work of one processor in half the time. However, there are a large number of problems to overcome in the use of parallel processing, not least the large body of people who have been constrained to solve problems in a serial fashion by present day computers. The demonstration of parallel programming in a wide variety of fields is therefore important. In some areas, this application of parallel techniques has been relatively straight forward; for example in graphics and signal processing. However, in general the problems of parallel processing are proving more difficult to solve and most applications must be individually researched. In this thesis, for example, parallel programming has been used in the field of Diesel engine simulation. Many of the techniques developed in this area may however be applicable in other areas of engineering simulation.

It is felt that the implementation of parallel processing techniques could lead to a similar dedication of computing resources to specific applications, as has happened in the case of the washing machine. A multi-microprocessor computer with the power of a super-computer would be relatively cheap and could be dedicated to a specific task. An example of this dedication, is the use of a large Transputer array for the cross referencing of police finger print evidence with the Home Office finger print records [2]. The use of a parallel computer has allowed two major improvements to the system: the finger print throughput of the computer has been greatly increased allowing several sets of finger print evidence to be referenced at one time, and it has allowed very quick referencing of important fingerprints through the dedication of the whole computer to one search. Possible future uses for a dedicated multi-processor computer, include its use in engine simulation for use in engine condition monitoring which is

described in this work.

1.3. The modern Diesel engine.

The development of the Diesel engine [3] has mirrored that of the computer. Since its invention in 1897, the Diesel engine has developed from its initial uses in large stationary plant to being the power source of choice in applications ranging from cars and trucks, to ships and power stations. Figure 1.1 shows an example of a large Diesel engine which is used for power generation on the island of Guernsey. The modern Diesel engine combines high efficiency with a high reliability. It also has the ability to burn low grade fuels. Diesel engines, however, have a higher initial capital cost and they are used where the saving in fuel consumption outweighs the initial cost.

The present trend in both Diesel and spark ignition engine development, is to increase the performance of the engine, whilst complying with tighter emission controls and improving the efficiency of the engine. These requirements are not always mutually compatible. A large number of new engine developments require the increased application of micro-electronics in engines. For example, Lucas and Emtage [4] discuss the use of hydrogen/petrol mixtures for burning in spark ignition engines. The use of hydrogen improves the lean burn characteristics of the engine improving both the performance and the emissions. However the problems of hydrogen storage make its use only practically possible at low loads. High loads have to be achieved using just petrol. The practical control of the proportion of petrol and hydrogen in such a system over its complete operating range is only possible with the use of microprocessor technology. Other new areas of engine control where the use of micro-electronics will be important are in the control of variable valve timing, variable geometry turbochargers [5] and

fuel injection [6]. The increased demands on engine performance are also leading to an increased need for component reliability. This is leading to increased monitoring of parts for wear which could adversely effect engine performance.

In future engine systems there will be a great increase in the number of microprocessor controlled and monitored engine parameters, such as those discussed above. Clark [7] discusses the many interactions that these controlled parameters have, such as the strong interaction between the variable geometry turbocharger restriction and the fuel injection timing. For optimal control of the complete engine he discusses the need for the coordination of all the control systems in the engine. This will require well defined communication standards between each of the engine control subsystems. A future engine control system could well be a parallel distributed computer.

1.4. An outline of the contents of this thesis.

Engine simulation will play a major part in the design and development of more advanced engines. However, the relative slowness of the more general thermodynamic engine models is a limitation to their use. In this thesis, the techniques of parallel processing have been applied to engine simulation. The engine simulation has been broken up into a number of segments which can be solved separately on different processors. Using this, an enhanced speed simulation has been produced. This work describes the development of this parallel Diesel engine simulation and its contents are arranged as follows:

Chapter 2 describes in detail the 'filling and emptying' method for engine simulation. This method has been used as basis for the parallel Diesel engine simulation developed in this thesis.

In **Chapter 3**, the parallel microprocessor computer and its operating system which have been used for the parallel Diesel engine simulation are outlined. The research for this thesis, on the parallel Diesel engine simulation, has shown some limitations to the performance of the present parallel computer system. The chapter describes these limitations and then outlines a possible new computer for the simulation of Diesel engines, which will overcome these limitations.

The parallel Diesel engine simulation developed for this thesis is described in **Chapter 4**. The chapter briefly describes the earlier implementation of a parallel Diesel engine simulation by Jones [8], with an analysis of the parallel computation structures used for this simulation. From this analysis, a totally new parallel computation structure for Diesel engine simulation has been designed and developed. The chapter describes this new simulation structure, which gives a simulation which is computationally more exact than the previous simulation whilst achieving a higher computation speed through the more efficient use of the parallel computer.

Chapter 5 is a description of the user interface to the new Diesel engine simulation developed in chapter 4.

In **Chapter 6**, possible methods of increasing the amount of parallelism in the Diesel engine simulation are discussed. The parallelism which is described in chapter 4, is limited by the type of engine which is simulated. New parallel methods will need to be developed to take advantage of the increased number of processing nodes in newer parallel computers. The chapter describes in detail the practical implementation of some new parallel methods for use in engine simulation.

Chapter 7 is a description of a Leyland TL11 experimental engine. This engine

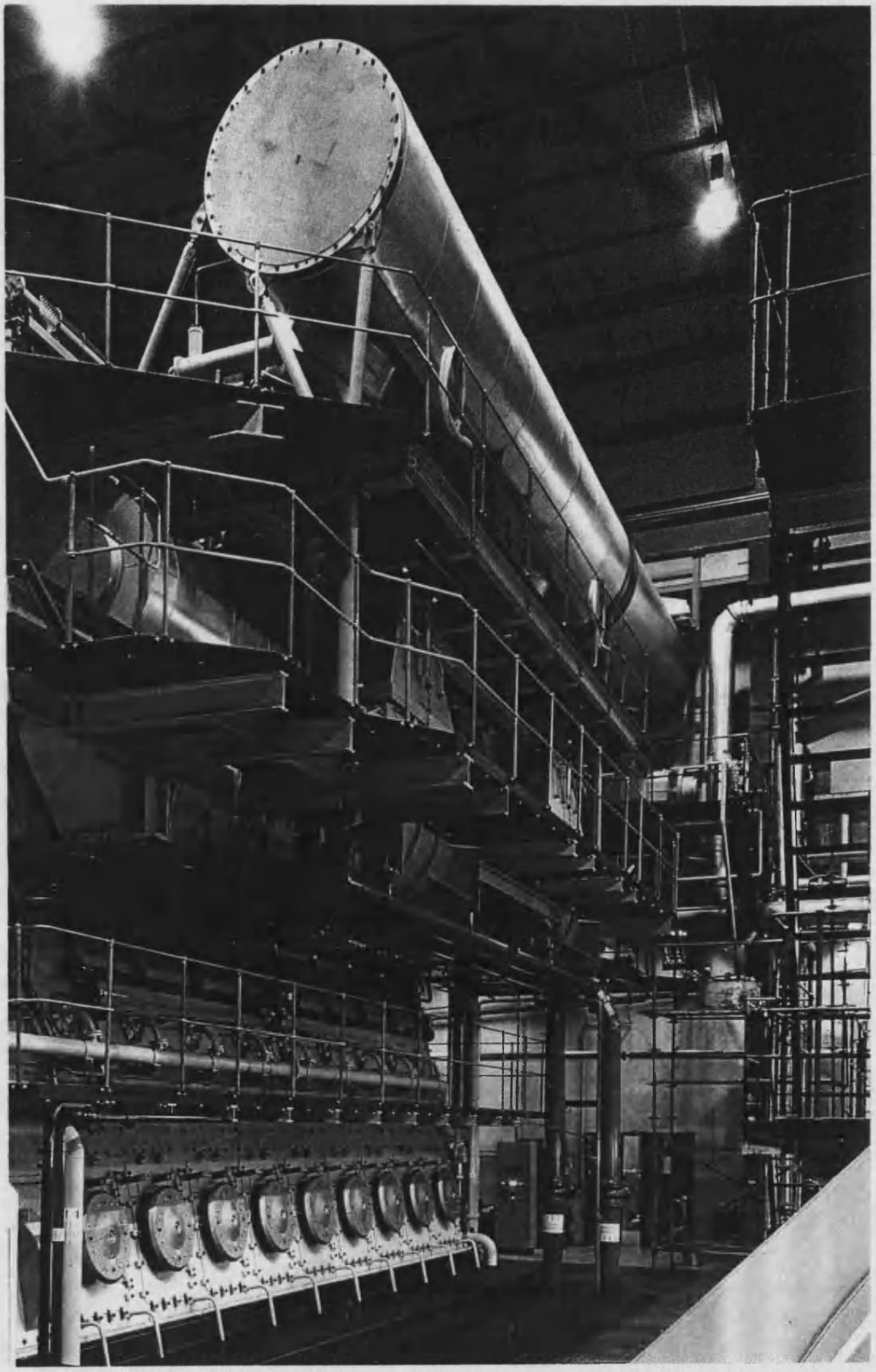
has been fully instrumented for the purpose of validating the parallel engine simulation in chapter 4. Chapter 7 also describes a new data acquisition computer, which has been developed for this thesis, for the recording of data from the engine.

In **Chapter 8** a comparison of the results from the experimental engine described in chapter 7 and the results from the parallel engine simulation described in chapter 4 is made.

Possible uses for a high performance simulation, such as the parallel simulation developed in this thesis, are discussed in **Chapter 9.**

The conclusions for this research are given in **Chapter 10** with suggestions for possible future work.

Figure 1.1. **A large Diesel engine used for electrical power generation on the Island of Guernsey.**
(courtesy of the States Electricity board (SEB), Guernsey)



CHAPTER 2

DIESEL ENGINE MODELLING.

2.1. Introduction.

Modelling has been used in Diesel engine design and development for a number of years. Models have been used to complement engine prototyping and instrumentation [9]. They allow an engine developer to get close to an optimum design before the prototype engine is built. This is cheaper as prototyping is costly. It is also quicker as computer models may more easily be adjusted and altered.

Diesel engine models vary considerably in their complexity and accuracy. They range from simple models based on linear equations to complex models based on the thermodynamic and fluid dynamic processes which occur within the engine. The simple models are limited in their application. This is either due to a reliance on a large amount of empirical data from an engine test bed in the case of simple physical models, or to small perturbations about an operating point for models derived from system identification. However the simple models are comparatively easy to solve and so can be calculated quickly. The complex Diesel engine models represent the physical processes within the engine and so can model an engine over its complete range. However the non-linear differential equations of the complex models are difficult to solve. They are therefore relatively slow to calculate on a serial computer.

The research in this thesis has been into the development of a complex Diesel engine model for a parallel computer. This chapter describes the complex Diesel engine model used for this research called the 'filling and emptying' method. It also briefly describes the limitations of this model and the possible enhancements to it that overcome these limitations.

2.2. The 'filling and emptying' method.

The 'filling and emptying' method derives its name from the manner in which it follows the successive 'filling and emptying' of the cylinder and manifolds of the engine with gas. The operation of the method shows a close resemblance to that of a real engine, this is as follows:

Air is introduced into the inlet manifold either directly from the atmosphere for a naturally aspirated engine, or through a compressor when the engine is turbocharged. Heat transfer occurs between the gas in the inlet manifold and the surroundings through the manifold wall. Air leaves the manifold through the cylinder inlet valves.

Air is introduced through the cylinder inlet valve when it is open from the engine inlet manifold. Exhaust gas leaves the cylinder through the cylinder exhaust valve when it is open to the exhaust manifold. Fuel is introduced into the cylinder when both valves are closed. Combustion results with heat and combustion products being released into the cylinder gases. The cylinder gases transfer work to and from the cylinder piston. Heat is transferred from the cylinder gases to the engine coolant through the cylinder wall.

Hot exhaust gas is introduced into the exhaust manifold through the cylinder exhaust valves. Heat is transferred from the exhaust manifold gases to the surroundings through the exhaust manifold wall. Exhaust gas leaves the manifold directly to atmosphere in the case of a naturally aspirated engine, or through a turbine if the engine is turbocharged.

The 'filling and emptying' method views a Diesel engine as a set of interconnected thermodynamic control volume models. Each of the control volumes represents one of the cylinder or manifold volumes of the engine. These control volumes are interconnected by valve and port models. The turbine and compressor are viewed as connections between the manifolds and the atmosphere with the flows through them being modelled by the compressor and turbine characteristics.

The 'filling and emptying' method makes the following assumptions in the modelling of an engine:

1. **Semi-perfect gas behaviour.**
   (a) The gas in the engine control volumes is assumed to satisfy the perfect gas relationship: $PV = mRT$.
   (b) The specific heats of the control volume vary with temperature and composition.
   (c) Gas dissociation effects are ignored and the thermodynamic properties of the gas are therefore a function of gas temperature and composition.

2. **Homogenous control volumes.**
   It is assumed that all control volumes contain a homogenous mixture and that any gas flows into a control volume are immediately mixed with the control volume gas, the so called perfect mixing assumption.

3. **Control volume thermodynamic equilibrium.**

   The contents of the engine control volumes are assumed to be in thermodynamic equilibrium such that one value of pressure and stagnation temperature fully defines the state of the control volume.

4. **Quasi steady state flow between control volumes.**

   The gas flows in a real Diesel engine are non-steady. The 'filling and emptying' method assumes that these non-steady flows can be modelled by quasi steady state flows. A quasi steady state flow is where the flow at any instant behaves as if it were steady.

The next section gives the derivation of the equations for the 'filling and emptying' method for a generalised engine control volume. The sections following this outline the changes to the generalised equation for the cylinder control volume, inlet manifold control volume and the exhaust manifold control volume. These sections also outline how the various terms in the control volume equations are calculated in the 'filling and emptying' method simulation of a Leyland TL11 truck engine described in chapter 7.

2.3. The derivation of the generalised 'filling and emptying' control volume equations.

Three equations describing the changes in control volume temperature, composition and trapped mass form the basis of the 'filling and emptying' method. The derivation of these three equations is given in the next three sections.

2.3.1. The variation of control volume temperature with respect to time.

The control volume temperature equation for the 'filling and emptying' method is based on the general energy equation for an open thermodynamic system. This is given in equation 2.1 with the effect of potential energy ignored.

$$\frac{dU}{dt} = \frac{dQ}{dt} - \frac{dW}{dt} + \sum_j \frac{dH_{0j}}{dt} \qquad \text{Equation 2.1.}$$

The general energy equation can be expanded to give equation 2.2.

$$m\frac{du}{dt} + u\frac{dm}{dt} = \sum_{sf} \frac{dQ_{sf}}{dt} - P\frac{dV}{dt} + \sum_j h_{0j}\frac{dm_j}{dt} \qquad \text{Equation 2.2.}$$

The internal energy u of the thermodynamic volume is a function of the local values of pressure, temperature and composition (fuel-air ratio). In the model used in this work the effects of chemical dissociation are ignored. Internal energy is therefore only a function of temperature and composition.

$$u = u(T, \lambda) \qquad \text{Equation 2.3a.}$$

$$\frac{du}{dt} = \left(\frac{\partial u}{\partial T}\frac{dT}{dt} + \frac{\partial u}{\partial \lambda}\frac{d\lambda}{dt}\right) \qquad \text{Equation 2.3b.}$$

Substituting equation 2.3b into the energy equation gives equation 2.4.

$$m\frac{\partial u}{\partial T}\frac{dT}{dt} + m\frac{\partial u}{\partial \lambda}\frac{d\lambda}{dt} + u\frac{dm}{dt} = -P\frac{dV}{dt} + \sum_{sf} \frac{dQ_{sf}}{dt} + \sum_j h_{0j}\frac{dm_j}{dt} \qquad \text{Equation 2.4.}$$

The enthalpy input into an engine control volume consists of two components: the enthalpy due to fuel burning within the volume and the enthalpy input due to flows through the junctions of the volume. This gives the variation of the control volume temperature with respect to time, equation 2.5.

$$\frac{dT}{dt} = \frac{1}{m\frac{\partial u}{\partial T}}\left(-p\frac{dV}{dt} + \sum_{sf}\frac{dQ_{sf}}{dt} + \sum_{j} h_{0j}\frac{dm_j}{dt} + h_{calv}\frac{dm_{fb}}{dt} - u\frac{dm}{dt} - m\frac{\partial u}{\partial \lambda}\frac{d\lambda}{dt}\right) \qquad \text{Equation 2.5.}$$

2.3.2. The variation of control volume trapped mass with respect to time.

Applying the principle of conservation of mass to the engine control volumes the variation of the control volume trapped mass with respect to time, equation 2.6 can be derived.

$$\frac{dm}{dt} = \sum_{j}\frac{dm}{dt}_{j} + \frac{dm_{fb}}{dt} \qquad \text{Equation 2.6.}$$

2.3.3. The variation of control volume composition with respect to time.

The composition of the engine control volumes varies as the proportion of fuel combustion products to air changes. This proportion is known as the fuel-air ratio. Fuel is only considered in the 'filling and emptying' method to be in the form of combustion products. In this section 'fuel' refers to the fuel combustion products. The ratio of the mass of 'fuel' to the mass of air in an engine volume is given by equations 2.7a and 2.7b.

$$\lambda = \frac{m_f}{m_{air}} \qquad \text{Equation 2.7a.}$$

$$m = m_f + m_{air} \qquad \text{Equation 2.7b.}$$

Differentiating this gives equation 2.8.

$$\frac{d\lambda}{dt} = \frac{m_{air}\frac{dm_f}{dt} - m_f\frac{dm_{air}}{dt}}{{m_{air}}^2} \qquad \text{Equation 2.8.}$$

Substituting for $m_{air} = m - m_f$ into this gives equation 2.9.

$$\frac{d\lambda}{dt} = \frac{1+\lambda}{m}\left[(1+\lambda)\frac{dm_f}{dt} - \lambda\frac{dm}{dt}\right] \qquad \text{Equation 2.9.}$$

The change in 'fuel' mass with respect to time is made up of all the 'fuel' flows through the junctions of the volume and the 'fuel' introduced through combustion in the volume. The mass of 'fuel' m_f in a mass of engine gas is given by equation 2.10a and may be differentiated to give equation 2.10b.

$$m_f = \frac{\lambda}{\lambda+1}m \qquad \text{Equation 2.10a.}$$

$$\frac{dm_f}{dt}_j = \frac{\lambda_j}{\lambda_j+1}\frac{dm_j}{dt} \qquad \text{Equation 2.10b.}$$

Substituting equation 2.10b into equation 2.9 and using a separate term for fuel introduced through burning gives equation 2.11 which is the generalised control volume equation for the variation of control volume composition (fuel-air ratio) with respect to time.

$$\frac{d\lambda}{dt} = \frac{1+\lambda}{m}\left[(1+\lambda)\left(\sum_j \frac{\lambda_j}{1+\lambda_j}\frac{dm}{dt}_j + \frac{dm_{fb}}{dt}\right) - \lambda\frac{dm}{dt}\right] \qquad \text{Equation 2.11.}$$

2.4. The cylinder control volume equations.

The general 'filling and emptying' method equations may be simplified for the six periods of the cylinder cycle: scavenge, induction, compression, combustion, power stroke, and exhaust. These equation simplifications are explained in the

next five sections.

A large number of variables in the engine cylinder control volumes are directly related to engine crankshaft angle. For this reason crankshaft angle rather than time has been used as the independent variable for the cylinder control volume equations. The relationship between engine crankshaft angle and time is given by equations 2.12a and 2.12b.

$$\theta = \omega_e \cdot t \qquad \text{Equation 2.12a.}$$

$$\frac{d}{d\theta} = \frac{1}{\omega_e} \cdot \frac{d}{dt} \qquad \text{Equation 2.12b.}$$

2.4.1. The cylinder scavenge period.

During the cylinder scavenge period the cylinder control volume has both its inlet and exhaust valves open. No combustion takes place in the volume and so the cylinder control volume equations for this part of the engine cycle are given in equations 2.13a, 2.13b and 2.13c.

$$\frac{dm}{d\theta}_{sc} = \frac{1}{\omega_e}\left(\frac{dm_i}{dt} - \frac{dm_e}{dt}\right) \qquad \text{Equation 2.13a.}$$

$$\frac{d\lambda}{d\theta}_{sc} = \frac{1+\lambda}{m}\left[\frac{(1+\lambda)}{\omega_e}\left(\frac{\lambda_i}{1+\lambda_i}\frac{dm_i}{dt} - \frac{\lambda_e}{1+\lambda_e}\frac{dm_e}{dt}\right) - \lambda\frac{dm}{d\theta}\right] \qquad \text{Equation 2.13b.}$$

$$\frac{dT}{d\theta}_{sc} = \frac{1}{m\frac{\partial u}{\partial T}}\left(-p\frac{dV}{d\theta} + \frac{1}{\omega_e}\left(\sum_{sf}\frac{dQ_{sf}}{dt} + h_{0i}\frac{dm_i}{dt} - h_{0e}\frac{dm_e}{dt}\right) - u\frac{dm}{d\theta} - m\frac{\partial u}{\partial \lambda}\frac{d\lambda}{d\theta}\right)$$

Equation 2.13c.

2.4.2. The cylinder induction period.

During the cylinder induction period the cylinder control volume has only its inlet valve open. No combustion takes place in the volume during the induction period. Two types of inlet valve flow are possible during the induction period. For normal flow, where the valve flow is from inlet manifold to cylinder, the cylinder control volume equations are given in equations 2.14a, 2.14b and 2.14c.

$$\frac{dm}{d\theta}_{in} = \frac{1}{\omega_e}\frac{dm_i}{dt} \qquad \text{Equation 2.14a.}$$

$$\frac{d\lambda}{d\theta}_{in} = \frac{1+\lambda}{m}\left[\frac{(1+\lambda)}{\omega_e}\left(\frac{\lambda_i}{1+\lambda_i}\frac{dm_i}{dt}\right) - \lambda\frac{dm}{d\theta}\right] \qquad \text{Equation 2.14b.}$$

$$\frac{dT}{d\theta}_{in} = \frac{1}{m\frac{\partial u}{\partial T}}\left(-p\frac{dV}{d\theta} + \frac{1}{\omega_e}\left(\sum_{sf}\frac{dQ_{sf}}{dt} + h_{0i}\frac{dm_i}{dt}\right) - u\frac{dm}{d\theta} - m\frac{\partial u}{\partial \lambda}\frac{d\lambda}{d\theta}\right)$$

Equation 2.14c.

When reverse flow occurs between the cylinder and the inlet manifold, then the rate of change in cylinder volume fuel-air ratio becomes zero.

2.4.3. The cylinder compression and power stroke periods.

During the compression and power stroke periods of the cylinder cycle, both the inlet and exhaust valves are closed. No combustion of fuel takes place in the cylinder control volume during these periods. The cylinder control volume equations, equations 2.15a, 2.15b and 2.15c, are therefore greatly simplified.

$$\frac{dm}{d\theta}_{cl} = 0 \qquad \text{Equation 2.15a.}$$

$$\frac{d\lambda}{d\theta}_{cl} = 0 \qquad \text{Equation 2.15b.}$$

$$\frac{dT}{d\theta}_{cl} = \frac{1}{m\frac{\partial u}{\partial T}}\left(-p\frac{dV}{d\theta} + \frac{1}{\omega_e}\sum_{sf}\frac{dQ_{sf}}{dt}\right)$$

Equation 2.15c.

2.4.4. The cylinder exhaust period.

During the exhaust period of the engine cycle the exhaust valve is open and the inlet valve is closed. Combustion does not take place in the volume during this period of the engine cycle. The exhaust valve flow is normally from the cylinder to the exhaust manifold. For normal valve flow the cylinder control volume equations are given by equations 2.16a. 2.16b and 2.16c.

$$\left(\frac{dm}{d\theta}\right)_{ex} = -\frac{1}{\omega_e}\frac{dm_e}{dt} \qquad \text{Equation 2.16a.}$$

$$\left(\frac{d\lambda}{d\theta}\right)_{ex} = 0 \qquad \text{Equation 2.16b.}$$

$$\left(\frac{dT}{d\theta}\right)_{ex} = \frac{1}{m\frac{\partial u}{\partial T}}\left(-p\frac{dV}{d\theta} + \frac{1}{\omega_e}\left(\sum_{sf}\frac{dQ_{sf}}{dt} - h_{0e}\frac{dm_e}{dt}\right) - u\frac{dm}{d\theta} - m\frac{\partial u}{\partial \lambda}\frac{d\lambda}{d\theta}\right)$$

Equation 2.16c.

However, when reverse flow does occur between the exhaust manifold and the cylinder volume, the equation for the cylinder fuel-air ratio is given by equation 2.16d.

$$\left(\frac{d\lambda}{d\theta}\right)_{ex} = \frac{1+\lambda}{m}\left[\frac{(1+\lambda)}{\omega_e}\left(-\frac{\lambda_e}{1+\lambda_e}\frac{dm_e}{dt}\right) - \lambda\frac{dm}{d\theta}\right] \qquad \text{Equation 2.16d.}$$

2.4.5. The cylinder combustion period.

During the combustion phase of the engine cycle both the inlet and the exhaust valve are closed. Mass, fuel and enthalpy are only introduced into the cylinder by fuel combustion. Fuel is considered liquid until combustion and therefore has no effect on the gas of the cylinder before combustion. The cylinder control volume equations during the combustion period are given by equations 2.17a, 2.17b and 2.17c.

$$\left(\frac{dm}{d\theta}\right)_{com} = \frac{dm_{fb}}{dt} \qquad \text{Equation 2.17a.}$$

$$\left(\frac{d\lambda}{d\theta}\right)_{com} = \frac{1+\lambda}{m\omega_e}\frac{dm_{fb}}{dt} \qquad \text{Equation 2.17b.}$$

$$\left(\frac{dT}{d\theta}\right)_{com} = \frac{1}{m\frac{\partial u}{\partial T}}\left(-p\frac{dV}{d\theta} + \frac{1}{\omega_e}\left(\sum_{sf}\frac{dQ_{sf}}{dt} + h_{calv}\frac{dm_{fb}}{dt}\right) - u\frac{dm}{d\theta} - m\frac{\partial u}{\partial \lambda}\frac{d\lambda}{d\theta}\right)$$

Equation 2.17c.

2.5. The manifold control volume equations.

The general 'filling and emptying' method equations may be simplified for the inlet and exhaust manifold control volumes. These equation simplifications are explained in the next two sections.

The manifold control volumes do not use cylinder crankshaft angle in their calculations. Therefore the manifold equations use time as their independent variable.

2.5.1. The inlet manifold control volume.

The inlet manifold has a constant volume and therefore the rate of change of volume with respect to time is zero. In the model used in this research the heat losses to the surface of the inlet manifold are ignored as these are small. In the TL11 engine model the inlet manifold is permanently connected to a compressor and intercooler at its inlet and to all the inlet valves of the engine cylinders at its outlet. The inlet manifold control volume equations are given by equations 2.18a, 2.18b and 2.18c.

$$\frac{dm}{dt}_{im} = \frac{dm_c}{dt} - \sum_i \frac{dm_i}{dt} \qquad \text{Equation 2.18a.}$$

$$\frac{d\lambda}{dt}_{im} = \frac{1+\lambda}{m}\left[(1+\lambda)\left(\frac{\lambda_c}{1+\lambda_c}\frac{dm_c}{dt} - \sum_i \frac{\lambda_i}{1+\lambda_i}\frac{dm_i}{dt}\right) - \lambda\frac{dm}{dt}\right] \qquad \text{Equation 2.18b.}$$

$$\frac{dT}{dt}_{im} = \frac{1}{m\frac{\partial u}{\partial T}}\left(h_{0c}\frac{dm_c}{dt} - \sum_i h_{0i}\frac{dm_i}{dt} - u\frac{dm}{dt} - m\frac{\partial u}{\partial \lambda}\frac{d\lambda}{dt}\right) \qquad \text{Equation 2.18c.}$$

2.5.2. The exhaust manifold control volume.

The exhaust manifold has a constant volume and therefore the rate of change of volume with respect to time is zero. In the TL11 engine model each of the exhaust manifolds is permanently connected to a turbine at its outlet and to half of the cylinder exhaust valves at its inlet. The exhaust manifold control volume equations are given by equations 2.19a, 2.19b and 2.19c.

$$\frac{dm}{dt}_{em} = \sum_e \frac{dm_e}{dt} - \frac{dm_t}{dt} \qquad \text{Equation 2.19a.}$$

$$\frac{d\lambda}{dt}_{em} = \frac{1+\lambda}{m}\left[(1+\lambda)\left(\sum_e \frac{\lambda_e}{1+\lambda_e}\frac{dm_e}{dt} - \frac{\lambda_t}{1+\lambda_t}\frac{dm_t}{dt}\right) - \lambda\frac{dm}{dt}\right] \qquad \text{Equation 2.19b.}$$

$$\frac{dT}{dt}_{em} = \frac{1}{m\frac{\partial u}{\partial T}}\left(\sum_{sf}\frac{dQ_{sf}}{dt} + \sum_e h_{0e}\frac{dm_e}{dt} - h_{0t}\frac{dm_t}{dt} - u\frac{dm}{dt} - m\frac{\partial u}{\partial \lambda}\frac{d\lambda}{dt}\right)$$

Equation 2.19c.

2.6. The calculation of the terms in the 'filling and emptying' method equations.

All the terms in the 'filling and emptying' method equations are either derived from physical laws or, where the process is not amenable to this, empirical equations are used. The following sections outline how these are calculated.

2.6.1. The rate of mass flow through the cylinder valves.

The rate of mass flow through a cylinder valve is an unsteady process. In the 'filling and emptying' method this flow is considered quasi steady state and one dimensional. Clearly the flow is not in fact one dimensional and so the secondary flow effects such as boundary layer separation and friction are taken account of by introducing an empirically based discharge coefficient C_{dis} into the calculation of valve flow area.

The one dimensional mass flow rate through the valve will be dependent on the pressure upstream of the valve flow P_u and the pressure at the valve throat P_v. The flow upstream of the valve throat is assumed to be isentropic. The pressure at the valve throat is unknown and therefore the assumption is made that no diffusion due to turbulent flow occurs between the valve throat and the downstream control volume. From this $P_v = P_d$ where P_d is the downstream volume pressure.

Two types of flow are possible through the cylinder valve: choked and non-choked. Choked flow will occur when the pressure ratio P_{ratio} across the cylinder valve is higher than the critical pressure ratio P_{crit} and sonic flow occurs.

$$P_{ratio} = \frac{P_u}{P_d} \qquad \text{Equation 2.20a.}$$

$$P_{crit} = \left(\frac{\gamma+1}{2}\right)^{\frac{\gamma}{\gamma-1}} \qquad \text{Equation 2.20b.}$$

The choked mass flow rate for the cylinder valve is given by equation 2.21.

$$\frac{dm_v}{dt}_{crit} = C_d A_v P_u \sqrt{\left[\frac{\gamma}{RT_u}\left(\frac{2}{\gamma+1}\right)^{(\gamma+1)/(\gamma-1)}\right]} \qquad \text{Equation 2.21.}$$

For non-choked flow, the mass flow rate for the cylinder valve is given by equation 2.22 where the valve flow is sub-sonic.

$$\frac{dm_v}{dt} = C_d A_v P_u \sqrt{\left\{\left(\frac{2\gamma}{\gamma-1}\right)\cdot\frac{1}{RT_u}\cdot\left(\left(\frac{P_d}{P_u}\right)^{2/\gamma} - \left(\frac{P_d}{P_u}\right)^{(\gamma+1)/\gamma}\right)\right\}} \qquad \text{Equation 2.22.}$$

Figure 2.1 gives the inlet and exhaust effective valve area $C_{dis}.A$ against engine crankshaft angle for the Leyland TL11 engine modelled in this research. Ideally these effective valve flow areas should be obtained from steady flow tests on the valve being used. However these flow tests are not available for the TL11 engine. They have therefore been calculated from a knowledge of the actual valve area and discharge coefficients estimated from data given by Garnish [10].

2.6.2. Heat transfer in the cylinder control volume.

Heat transfer in the cylinder control volume to the cylinder wall is due to convection and radiation. Heat transfer due to conduction is assumed to be negligible. The heat transfer is therefore described by equation 2.23a.

$$\frac{dQ_w}{dt} = A \cdot h(T - T_{sf}) + A \cdot \epsilon\sigma(T - T_{sf}) \qquad \text{Equation 2.23a.}$$

In this work the contribution to heat transfer due to radiation in the cylinder volume is ignored. Researchers disagree on the contribution of radiation to the heat transfer in the cylinder. This is due to the difficulty in separating the contribution from both forms of heat transfer in experimental results. The simplified heat transfer model used in this research is therefore given by equation 2.23b.

$$\frac{dQ_w}{dt}_{cyl} = A_{cyl} \cdot h(T - T_{sf}) \qquad \text{Equation 2.23b.}$$

2.6.2.1. The calculation of the heat transfer coefficient *h*.

For the cylinder control volume the calculation of the heat transfer coefficient h is difficult. This coefficient depends on the flow conditions near the cylinder surfaces. The flows within an engine cylinder are very complex, with flows entering and leaving the cylinder through the valves, combustion occurring and the cylinder piston moving. Therefore a mathematical derivation of h is impossible and a semi-empirical heat transfer model has to be used. The semi-empirical heat transfer model used in this research is that due to Hohenburg [11]. The Hohenburg model assumes that the heat transfer in the cylinder is totally convective in nature. The model relates the heat transfer coefficient to

the cylinder temperature, pressure and volume as well as the mean value of piston speed. The Hohenburg heat transfer coefficient model is given by equation 2.24.

$$h = \frac{C_1 P^{0.8}(\overline{v_{pis}} + C_2)^{0.8}}{V^{0.06} T^{0.4}}$$

$$\overline{v_{pis}} = \frac{2r\omega_e}{\pi} \qquad \text{Equation 2.24.}$$

Ideally the constants C_1 and C_2 should be determined from heat transfer measurements on the engine being modelled. However for the TL11 engine model the constants given by Hohenburg have been used.

$$C_1 = 0.013,\ C_2 = 1.4$$

2.6.2.2. The calculation of the cylinder surface area *A* for the heat transfer calculation.

The surface area of the cylinder control volume is calculated from a knowledge of the cylinder geometry. The geometry of the cylinder is shown in figure 2.2. The surface area of the cylinder is given by equation 2.25.

$$A_{cyl} = A_{TDC} + \pi d \left(l + r(1 - \cos\theta) - \sqrt{l - r^2 \sin^2\theta} \right) \qquad \text{Equation 2.25.}$$

The 'filling and emptying' method assumes thermodynamic equilibrium. The temperature and the velocity of the gas in the vicinity of the piston land is in fact lower than for the rest of the volume. The Hohenburg model therefore over-estimates the heat transfer due to the area of the piston land. Hohenburg [11] suggests that the area of the piston land used in the calculation of heat transfer is altered to take account of this. He suggests a value for piston land area equal to the real land area raised to the arbitrary power of 0.3. Because of

its arbitrary nature, this adjustment factor has not been used in the research for this thesis.

2.6.2.3. The calculation of the cylinder surface temperature T_{sf}.

The heat transfer from the cylinder through the combustion wall to the engine coolant is represented by the simple heat transfer model given in figure 2.3. The thermal resistance R_1 is calculate from the heat transfer coefficient h derived in section 2.6.2.1. The thermal resistance R_2 and capacitance C represent the heat transfer model for the heat flow from the cylinder wall to the engine coolant. From this model the rate of change in cylinder wall temperature, equation 2.26, can be calculated.

$$\left(\frac{dT_{sf}}{dt}\right)_{cyl} = \frac{1}{C}\frac{dQ_w}{dt} - \frac{T_{cyl} - T_{sf}}{C \cdot R_2} \qquad \text{Equation 2.26.}$$

2.6.3. Heat transfer in the exhaust manifold control volume.

Heat transfer in the manifold is represented by the same one dimensional heat transfer model as for the cylinder. This model is shown in figure 2.3. The heat transfer between the manifold gas and the manifold wall is presumed to be totally convective in nature. The rate of heat transfer is therefore given by equation 2.27.

$$\left(\frac{dQ_w}{dt}\right)_{man} = A_{man} \cdot h\,(T - T_{sf}) \qquad \text{Equation 2.27.}$$

The thermal resistance R_2 and the thermal capacitance C models the conduction between the manifold wall and the surroundings. The rate of change of the manifold wall temperature with time is given by equation 2.28.

$$\left(\frac{dT_{sf}}{dt}\right)_{man} = \frac{1}{C}\frac{dQ_w}{dt} - \frac{T_{man} - T_{sf}}{C \cdot R_2} \qquad \text{Equation 2.28.}$$

2.6.4. Combustion in the cylinder control volume.

The pattern of fuel burning in the cylinder control volume changes as combustion takes place. There are three distinct phases to combustion and these are illustrated in figure 2.4. Firstly there is the ignition delay where the part of the fuel vaporises and mixes with the air in the cylinder. This is followed by the rapid combustion of the vaporised fuel and air mixture which is known as 'pre-mixed' burning. Finally there is a slower phase of combustion which is determined by the available oxygen in the combustion chamber and the mixing rate. This is known as 'diffusion' burning.

2.6.4.1. The calculation of cylinder ignition delay.

On being introduced into the engine cylinder, the fuel does not immediately ignite. The time between the injection of fuel and combustion is known as the ignition delay. This delay time is calculated in this thesis using the semi-empirical ignition delay formula due to Wolfer [12] with coefficients suggested by Watson [13]. The ignition delay formula is given in equation 2.29. It is based on the mean values of pressure and temperature between the injection of fuel and the ignition of the fuel.

$$t_{id} = \frac{453.4638}{\overline{P}^{1.022}} e^{\frac{2100}{T}} \qquad \text{Equation 2.29.}$$

2.6.4.2. The calculation of fuel burning rate in the cylinder control volume.

The model used for the fuel burning rate in this research is that due to Watson et al. [14]. During the ignition delay period the fuel burning rate is zero. After ignition the fuel burning rate is made up of two components: fuel burning due to 'pre-mixed' burning and due to 'diffusion' burning. The equation for this is given in equation 2.30.

$$FBR(\tau) = \beta f_1(\tau) + (1-\beta) f_2(\tau) \qquad \text{Equation 2.30.}$$

The 'mode of burning' factor β defines the proportion of the injected fuel which is burnt in the 'pre-mixed' form of burning. The equation for the rate of 'pre-mixed' burning is given by equation 2.31.

$$f_1(t) = 1 - (1 - t^{C_1})^{C_2} \qquad \text{Equation 2.31.}$$

The equation for the rate of 'diffusion' burning is given by equation 2.32.

$$f_2(t) = 1 - e^{(-C_3 t^{C_4})} \qquad \text{Equation 2.32.}$$

Both of these equations represent the shape of the fuel burning rate function and so are non-dimensionalised with respect to the amount of fuel injected into the cylinder and the nominal time of combustion. The combustion equation variables *C1, C2, C3, C4* and β are dependent on the engine operating condition. For the research in this thesis these values have been taken from the results obtained by Watson [14] for a Leyland truck engine. These are given in equations 2.33a, 2.33b, 2.33c, 2.33d and 2.33e.

$$C_1 = 2.0 + 44.549(t_{id}\omega_e)^{2.4} \qquad \text{Equation 2.33a.}$$

$$C_2 = 5000 \qquad \text{Equation 2.33b.}$$

$$C_3 = \frac{2.5463}{\lambda^{0.644}} \qquad \text{Equation 2.33c.}$$

$$C_4 = 0.791 {C_3}^{0.248} \qquad \text{Equation 2.33d.}$$

$$\beta = \frac{1 - 0.4125\lambda^{0.37}}{{t_{id}}^{0.26}} \qquad \text{Equation 2.33e.}$$

From the non-dimensionalised fuel burning rate equation 2.30, the actual rate of fuel burning with respect to engine crankshaft angle can be derived. This is given in equation 2.34.

$$\frac{\mathrm{d}m_{fb}}{\mathrm{d}\theta} = \frac{FBR(\tau) \cdot m_f}{\theta_{comb}} \qquad \text{Equation 2.34.}$$

The nominal fuel burning duration has been taken as 125^{o} for this research.

2.6.5. The calculation of the volume of the cylinder, and rate of change of volume.

The volume of the cylinder and the rate of change of the cylinder volume are derived from the geometry of the cylinder and the cylinder piston. A diagram of this geometry is given in figure 2.3. The equation for the volume of the cylinder is given by equation 2.35. The term in equation 2.35 based on the engine compression ratio C_{ratio} gives the volume of the cylinder at piston top dead centre (TDC).

$$V_{cyl} = \frac{\pi d^2}{4}\left[l + r(1 - \cos\theta) - \sqrt{l^2 - r^2\sin^2\theta} + \frac{2r}{\Gamma - 1}\right] \qquad \text{Equation 2.35.}$$

Differentiating equation 2.35 with respect to crankshaft angle gives the rate of change of cylinder volume with respect to the engine crankshaft angle, equation 2.36.

$$\frac{dV_{cyl}}{d\theta} = \frac{\pi d^2}{4}\left(r\sin\theta + \frac{r^2\sin\theta\cos\theta}{\sqrt{l^2 - r^2\sin^2\theta}}\right) \qquad \text{Equation 2.36.}$$

2.6.6. The engine gas properties model.

The internal energy u of the all the engine control volumes is a function of temperature and composition. The effect of pressure is ignored as the model ignores the effects of dissociation. The gas constant R is a function of composition (fuel-air ratio) only. Tabulated values for the internal energy and the gas constant for the equilibrium combustion products of a hydrocarbon (C_nH_{2n}) and air have been derived by Newall and Starkman [15]. From these Charlton [16] has derived approximate algebraic expressions for the internal energy, the gas constant, and the rate of change of internal energy with respect to fuel-air ratio and temperature. These expressions are for lean fuel-air ratio mixtures and are given in equations 2.37a, 2.37b 2.37c, 2.38, 2.39a, 2.39b 2.39c and 2.39d

$$u = \frac{K_a(T) - K_b(T)\lambda}{1+\lambda} \qquad \text{Equation 2.37a.}$$

$$K_a = 696.0T + 0.89465\times10^{-3}T^2 + 102.512\times10^{-6}T^3 - 45.52\times10^{-9}T^4 + 6.22\times10^{-12}T^5 \qquad \text{Equation 2.37b.}$$

$$K_b = 722.903T + 1.57193T^2 - 523.84\times10^{-6}T^{-6} + 116.61\times10^{-9}T^4 - 12.5335\times10^{-12}T^5 \qquad \text{Equation 2.37c.}$$

$$R = \frac{270.21 + 288.294\lambda}{1+\lambda} \qquad \text{Equation 2.38.}$$

$$\frac{\partial u}{\partial T} = (101.4663 - 2.52237T - 1.055 \times 10^{-3}T^2 + 1.7569 \times 10^{-7}T^3)\,\lambda$$

$$+920.7128 - 0.2806T - 5.4939 \times 10^{-5}T^2 - R \qquad \text{Equation 2.39a.}$$

$$\frac{\partial u}{\partial \lambda} = \frac{26.053T + 1.57104T^2 - 626.352 \times 10^{-6}T^3 + 162.1283 \times 10^{-9}T^4 - 18.7535 \times 10^{-12}T^5}{(1+\lambda)^2}$$

Equation 2.39b.

The specific stagnation enthalpy of the volume gas is given by equation 2.40 and the ratio of specific heats for the volume is given by equation 2.41.

$$h_0 = u + RT \qquad \text{Equation 2.40.}$$

$$\gamma = 1 + \frac{R}{\frac{\partial u}{\partial T}} \qquad \text{Equation 2.41.}$$

It should be noted that in the simulation developed by Jones [8] the algebraic expressions for the gas properties were those due to Krieger et al. [17]. These were changed, in the simulation in this thesis, in order that energy balance calculations could be performed on each engine control volume.

2.6.7. The turbocharger model.

The Leyland TL11 engine is fitted with a turbocharger. This supplies air to the inlet manifold through a centrifugal compressor and an intercooler. The compressor is driven through a shaft by a divided entry variable geometry turbine through which the gases from the two engine exhaust manifolds flow. The models for both the turbine and the compressor flows are based on their steady state flow parameters which are obtained from experimental measurement. The non-steady flows through both the turbine and the compressor are therefore modelled as quasi steady state flows.

The next two sections outline in detail the models for the compressor and the turbine.

2.6.7.1. The turbine model.

The TL11 turbine is modelled as two imaginary turbines through which the exhaust gases from each exhaust manifold flow. The turbines are imagined to be connected to the same shaft and to be rotating at the same speed.

Holset, the manufacturers of the turbine, were unable to provide the performance map for the variable geometry turbine and so an alternative flow model has had to be used. The simpler turbine model is based on the swallowing curve measured by Roberts [18]. This swallowing curve is given in figure 2.5. The swallowing curve was obtained by running the engine at its limiting torque with the turbine restriction fully open. The model therefore calculates the same mass flow rate for all turbine speeds. The turbine mass flow rate is given by equation 2.42 where the function $f(P_r)$ is taken from the turbine swallowing curve. The turbine restriction is modelled as a scaling factor on the turbine mass flow rate. This scaling factor reduces the flow rate linearly with turbine restriction. This linear relationship between mass flow rate and turbine restriction was shown by Roberts. The turbine measurements shown in figure 2.5 are for the complete turbine flow. The flow through each of the two imaginary turbines therefore has to be halved.

$$\frac{dm_t}{dt} = 1 - 0.8 \cdot restriction \cdot \frac{f(P_{ratio})\,P_{ratio}}{2\sqrt{T_{tin}}} \qquad \text{Equation 2.42.}$$

The temperature of the exhaust gas leaving the turbine can be determined from a knowledge of the turbine isentropic efficiency and expansion ratio. This is defined as the ratio between actual turbine work to isentropic turbine work. The temperature at the turbine exit is given by equation 2.43.

$$T_{texit} = T_{tin} - \eta_t T_{tin}\left(1 - \frac{P_{tin}}{P_{texit}}^{\frac{(1-\gamma)}{\gamma}}\right) \qquad \text{Equation 2.43.}$$

For the model used in this research the turbine isentropic efficiency has been taken as a constant value of 0.5. The turbine torque for each imaginary turbine can be calculated from the turbine exit flow temperature and this is given in equation 2.44.

$$\tau_t = \frac{\mathrm{d}m_t}{\mathrm{d}t}\frac{\Delta h_0}{\omega_{tc}} \qquad \text{Equation 2.44.}$$

2.6.7.2. The compressor and intercooler model.

The compressor steady state performance map is shown in figure 2.6. Given the pressure ratio across the compressor, the turbine rotational speed, the upstream temperature and pressure the mass flow through the compressor can be calculated. The compressor efficiency may also be determined from the steady state performance map. Using this, the compressor outlet temperature, mass flow and enthalpy may be determined. The compressor outlet temperature is calculated using equation 2.45.

$$T_{cexit} = T_{cin} - \frac{T_{cin}}{\eta_c}\left(1 - \frac{P_{cin}}{P_{cexit}}^{\frac{(1-\gamma)}{\gamma}}\right) \qquad \text{Equation 2.45.}$$

The torque required by the compressor can be calculated based on the change of enthalpy of the gas through the compressor. This is given in equation 2.46.

$$\tau_c = \frac{\mathrm{d}\, m_c}{\mathrm{d}t} \frac{\Delta h_0}{\omega_{tc}} \qquad \text{Equation 2.46.}$$

The gas from the compressor passes through an inter-cooler prior to entry into the inlet manifold. The inter-cooler removes heat from the compressed gas thus increasing the density of the gas in the inlet manifold. The effectiveness of this cooling is given by equation 2.47. This relates the actual temperature drop across the inter-cooler to the theoretical temperature drop due to the cooling water.

$$\eta_{ic} = \frac{T_{in} - T_{out}}{T_{in} - T_{coolant}} \qquad \text{Equation 2.47.}$$

In the research given in this thesis, a constant value of intercooler effectiveness of 0.9 has been used. From this, the inter-cooler exit temperature can be calculated using equation 2.48.

$$T_{out} = (1 - \eta_{ic}) T_{in} + \eta_{ic} T_{coolant} \qquad \text{Equation 2.48.}$$

The intercooler also causes a pressure drop due to the resistance offered by the cooling surfaces. For the Diesel engine model, this intercooler pressure drop has been made a constant of 10,000Pa. This pressure drop was determined by experimental measurement on the TL11 engine described in chapter 7. The pressure ratio used in the calculation of the compressor mass flow is therefore given by equation 2.49.

$$P_{ratio} = \frac{P_{im} + \Delta P_{ic}}{P_{ambient}} \qquad \text{Equation 2.49.}$$

2.7. The dynamic engine model.

Using the model already described, the steady state operation of the TL11 engine can be simulated. This type of simulation would use constant values for parameters such as engine speed, injection timing and mass of fuel injected per cycle. By incorporating models which describe the dynamics of the engine into the 'filling and emptying' method, the dynamic response of the engine can be modelled. The dynamic models used to simulate the TL11 engine are described in the following sections.

2.7.1. The engine crankshaft dynamics.

All the cylinders in the engine model are connected together through the engine crankshaft. The dynamics of this crankshaft are determined by the torque generated by the cylinders, the demanded engine torque, the inertia of the crankshaft and the inertia of the engine load.

The torque generated by the cylinders is due to the pressure of the cylinder gas acting on the cylinder piston. This indicated cylinder torque is given in equation 2.50.

$$\tau_{cyl} = (P_{cyl} - P_{ambient}) \frac{\mathrm{d}V_{cyl}}{\mathrm{d}\theta} \qquad \text{Equation 2.50.}$$

However not all of the indicated torque is transferred to the engine crankshaft. Some is lost due to engine friction. This engine friction consists of two parts: firstly the genuine friction losses which occur between the cylinder sliding surfaces and in the crankshaft bearings and secondly there are the losses due to driving engine auxiliaries such as the water and oil pumps.

Engine friction is normally determined by engine motoring tests. In these tests, the power required to rotate the engine is measured. Both Millington and Hartles [19] and Chen and Flynn [20] have proposed models for engine friction based on motoring tests. The first of these models was for a naturally aspirated engine and the second was for a turbocharged engine. As the engine used in this research is a turbocharged engine, the Chen and Flynn friction model has been used. The model is given in equation 2.51 and is based on the mean piston speed and the maximum cylinder pressure for each engine cycle.

$$FMEP = C_1 + C_2 P_{max} + C_3 \overline{v_{pis}} \qquad \text{Equation 2.51.}$$

The result of the friction calculation is the cylinder friction mean effective pressure (FMEP). The constants in the Chen and Flynn friction model should ideally be obtained from motoring experiments on the engine being modelled. However, for this research, the constants proposed by Chen and Flynn have been used.

$$C_1 = 13{,}700 \quad C_2 = 0.005 \quad C_3 = 16{,}200$$

The brake cylinder torque generated by each cylinder in the engine, which takes account of the engine friction, is given by equation 2.52.

$$\tau_{cyl} = (P_{cyl} - P_{ambient}) \frac{dV_{cyl}}{d\theta} - \left| \left(FMEP \cdot \frac{dV_{cyl}}{d\theta} \right) \right| \qquad \text{Equation 2.52.}$$

Using the sum of the brake cylinder torques from all of the cylinders, the engine crankshaft acceleration can be calculated using the equation 2.53.

$$\frac{d\omega_e}{dt} = \frac{\sum \tau_{cyl} - \tau_{load}}{J_e + J_{load}} \qquad \text{Equation 2.53.}$$

2.7.2. The turbocharger crankshaft dynamics.

For the model used in this research, the turbocharger rotor has been assumed to have a friction load which is proportional to the turbine torque.. Therefore the turbocharger crankshaft acceleration is based on the torque demanded by the compressor, the friction adjusted torques produced by the two imaginary turbines and the turbocharger shaft inertia. The turbocharger acceleration is given in equation 2.54.

$$\frac{d\omega_{tc}}{dt} = \frac{\eta_{tc}\sum\tau_t - \tau_c}{J_{tc}} \qquad \text{Equation 2.54.}$$

2.7.3. The engine control actuator dynamics.

The TL11 engine has three control inputs. These control the fuel rack position, the timing of the fuel injection, and the turbine nozzle restriction. All of the engine controls are driven by hydraulic actuators in a closed control loop. Figure 2.7 shows a schematic diagram of the engine hydraulic actuators.

Jones [8] modelled the control actuators using the results from a system identification on the actuators. System identification uses a pseudo random binary sequences (PRBS) as a control input. Jones found that the hydraulic actuator dynamics were predominantly second order in nature, with a velocity limit due to a maximum limit to the hydraulic fluid flow. The general equations for the engine actuators are given in equations 2.55a and 2.55b. Table 2.1 gives the equation values for each of the control actuators.

$$\dot{x}_{actuator} < \dot{x}_{limit}$$

$$\frac{x_{actuator}}{x_{demand}} = \frac{k\omega_n{}^2}{s^2 + 2\xi\omega_n s + \omega_n{}^2} \qquad \text{Equation 2.55a.}$$

$$\dot{x}_{actuator} > \dot{x}_{limit}$$

$$\dot{x}_{actuator} = \dot{x}_{limit} \qquad \text{Equation 2.55b.}$$

TABLE 2.1. The TL11 control actuator model constants.

Actuator	Gain K	Damping ratio ξ	Natural frequency ω_n (rad/s)	Slew rate (m/s)
Fuel rack	0.98	0.55	174	0.1
Turbine nozzle	1.05	0.76	142	0.8
Fuel injection	1.0	0.7	125	0.2

2.7.4. The engine fuel delivery system.

Combustion in the cylinders is determined by the amount of fuel injected into the cylinder and the time at which it is injected. These are determined by the fuel delivery system which is controlled by the actuators outlined above. The fuel delivery system consists of the fuel pump, the fuel delivery pipes and the fuel injectors. These are modelled for this research in terms of the inputs that are needed for the combustion model described in section 2.6.4. These model parameters are the crankshaft angle at which the fuel is injected into the cylinder which is known as the dynamic timing, and the mass of fuel injected into the cylinder.

The fuel delivery system has been modelled using its steady state performance characteristic and a fuel transport delay which represents the time delay for the fuel travelling down the fuel delivery pipes, The steady state performance characteristic for the fuel delivery system is given in figure 2.8. The mass of

fuel injected is a function of the rack position determined by the actuator control equations and the engine speed. The dynamic injection timing is given by equation 2.56. The dynamic timing is calculated from the static injection timing, which is controlled by the injection actuator model, and the fuel transport delay. The fuel is presumed to travel at the same speed as the fuel pressure wave down the fuel delivery pipe. The fuel pressure wave propagates at the speed of sound (C).

$$\theta_d = \theta_s + \frac{l_{pipe}\omega_e}{C} \qquad \text{Equation 2.56.}$$

2.8. Advanced Diesel engine modelling techniques.

Although the 'filling and emptying' model gives a very detailed representation of a Diesel engine, it still has a number of limitations. In this section these limitations are discussed and modelling techniques which overcome these are briefly outlined.

2.8.1. The method of Characteristics.

The 'filling and emptying' method assumes thermodynamic equilibrium in each of the control volumes. This is a reasonable assumption in the cylinder (except during combustion) and inlet manifold control volume. However, for the exhaust manifold the spatial variation of pressure and temperature can be important. This is particularly true for pulse turbocharging systems where the effect of pressure pulses in the manifold is important or in long manifolds where the time delay from reflected pressure pulse from the end of the manifold is high.

If the effects of pressure wave action need to be taken account of, then the equations for compressible, unsteady flow must be solved. These may be solved using the method of characteristics. This is a method for solving the hyperbolic partial differential equations which the compressible flow equations are. The method is described by Benson et al. [21].

When modelling using the method of characteristics, it is usual to model the remaining engine control volumes using the 'filling and emptying' method. It is estimated that the use of the method of characteristics increases the time needed to compute an engine solution by about ten times.

2.8.2. Multi-zone cylinder models.

The need to model the emissions from an engine is becoming increasingly important. The 'filling and emptying' model assumes that the cylinder control volumes are homogenous and in thermodynamic equilibrium. For a study of emissions it is important to model the local variations within the cylinder volume during combustion and whilst the fuel is being injected. One example of a multi-zone cylinder model is described by Megerdichian et al. [22] which takes account of the variations in gas composition in the cylinder.

Figure 2.1. The inlet and exhaust effective valve area against engine crankshaft angle for the Leyland TL11 engine.

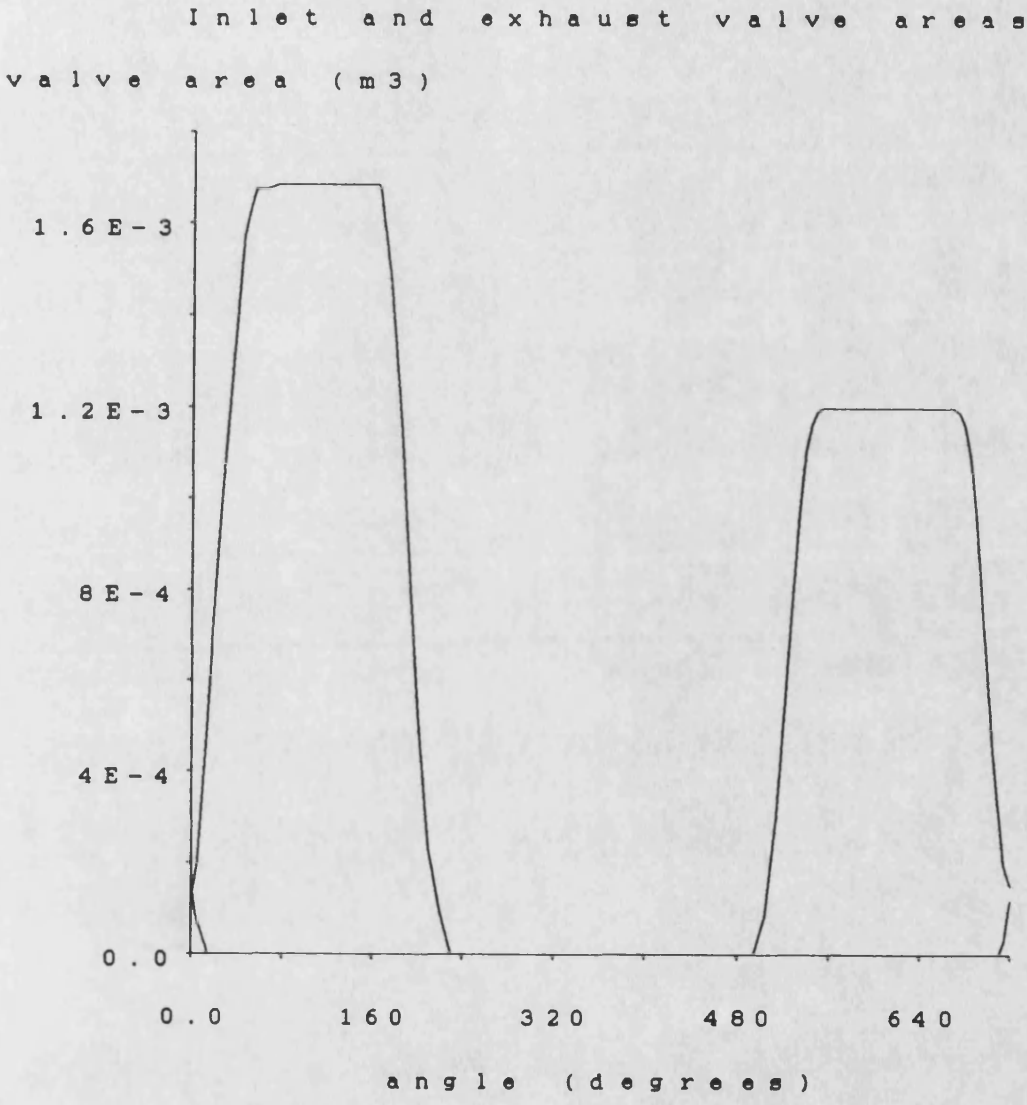


Figure 2.2 **The Leyland TL11 cylinder geometry.**

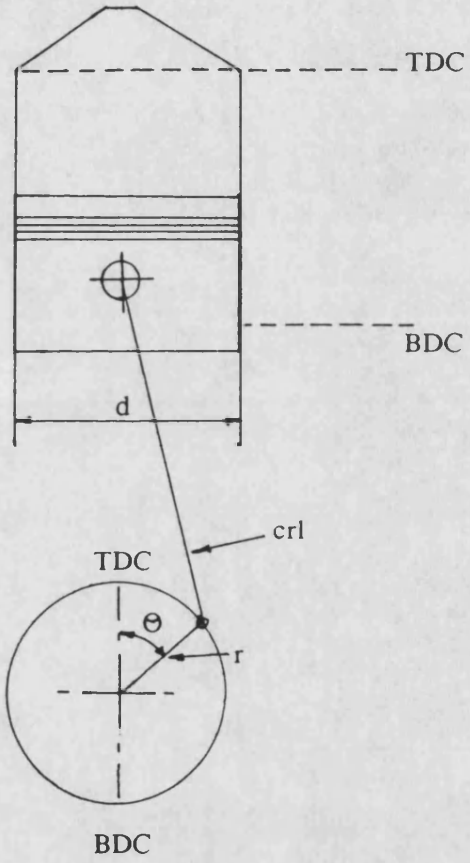


Figure 2.3. **The cylinder and manifold control volume heat transfer model.**

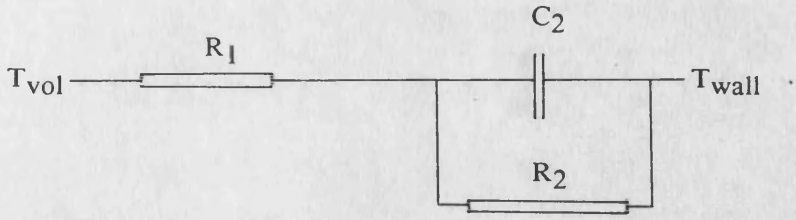


Figure 2.4a **The phases of combustion in the cylinder.**

1. Ignition delay.
2. Pre-mixed burning phase.
3. Diffusion burning phase.
4. Combustion tail.

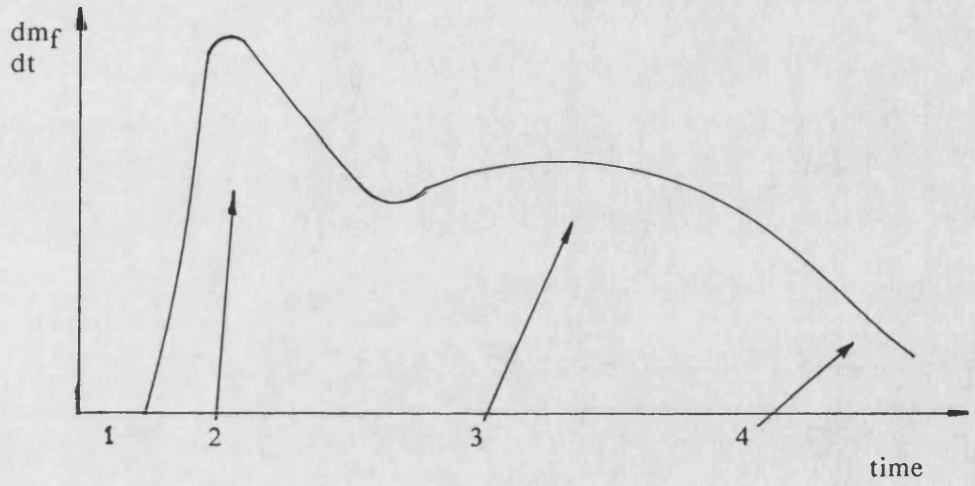


Figure 2.4b **The phases of the cylinder combustion model.**

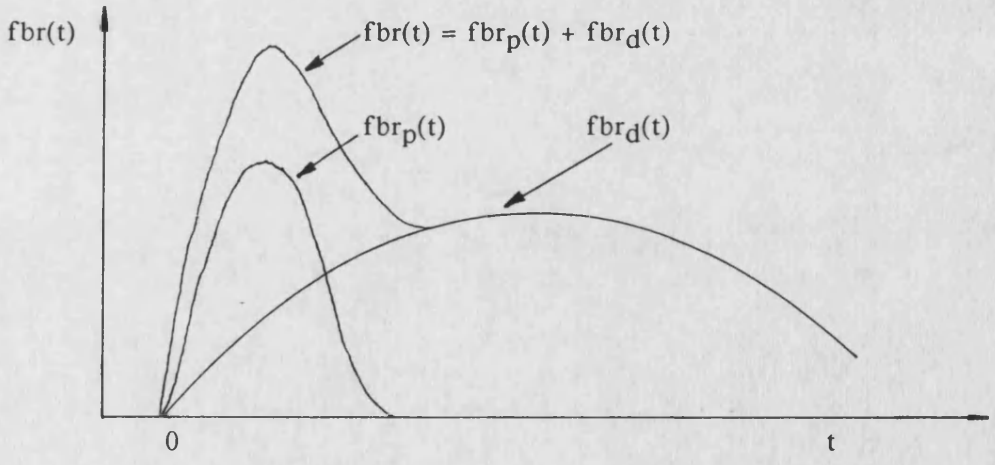


Figure 2.5 **The Holset turbocharger swallowing curve for full load with a 0% outlet restriction.**

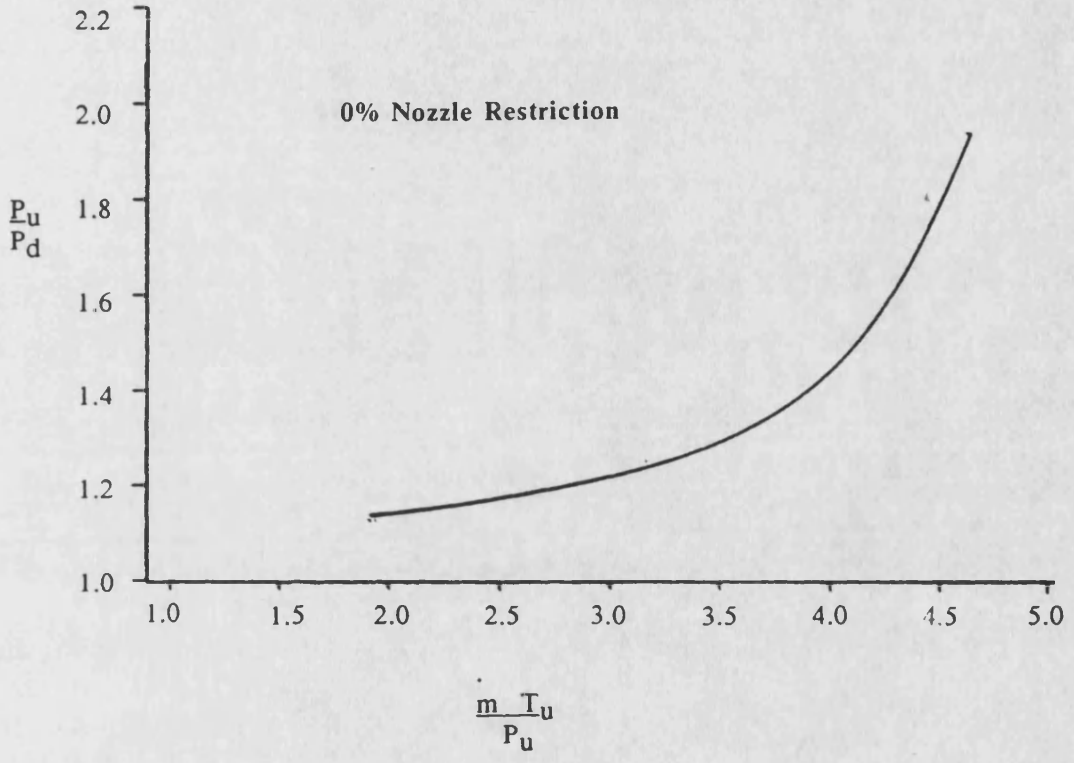


Figure 2.6 **The compressor steady state characteristic map.**

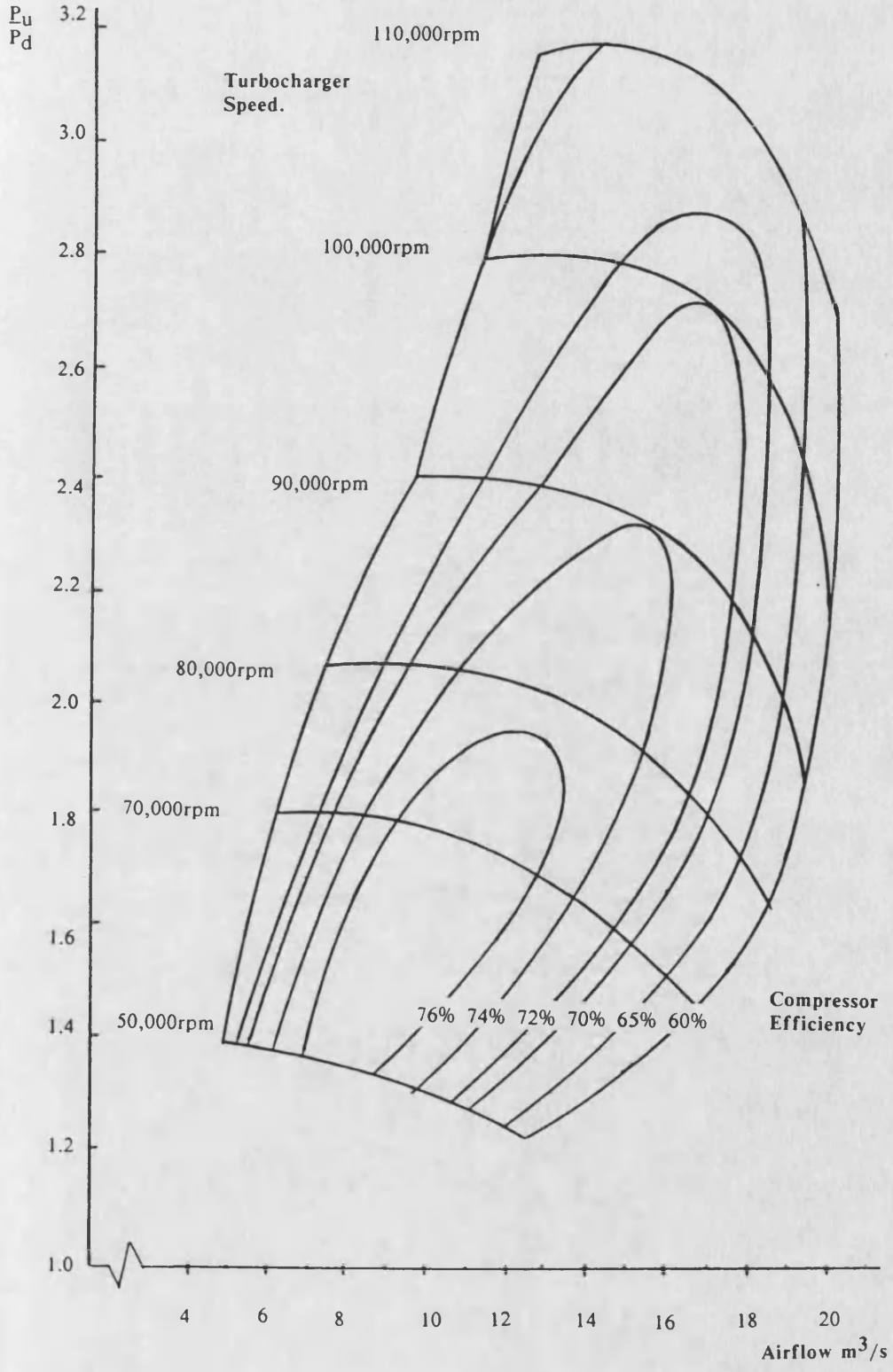


Figure 2.7 **A schematic diagram of the engine hydraulic actuators.**

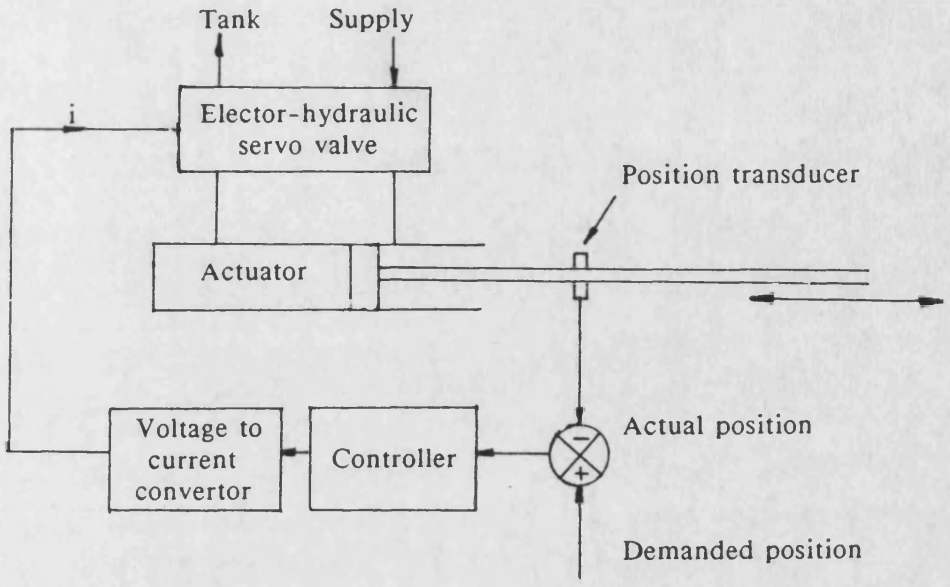
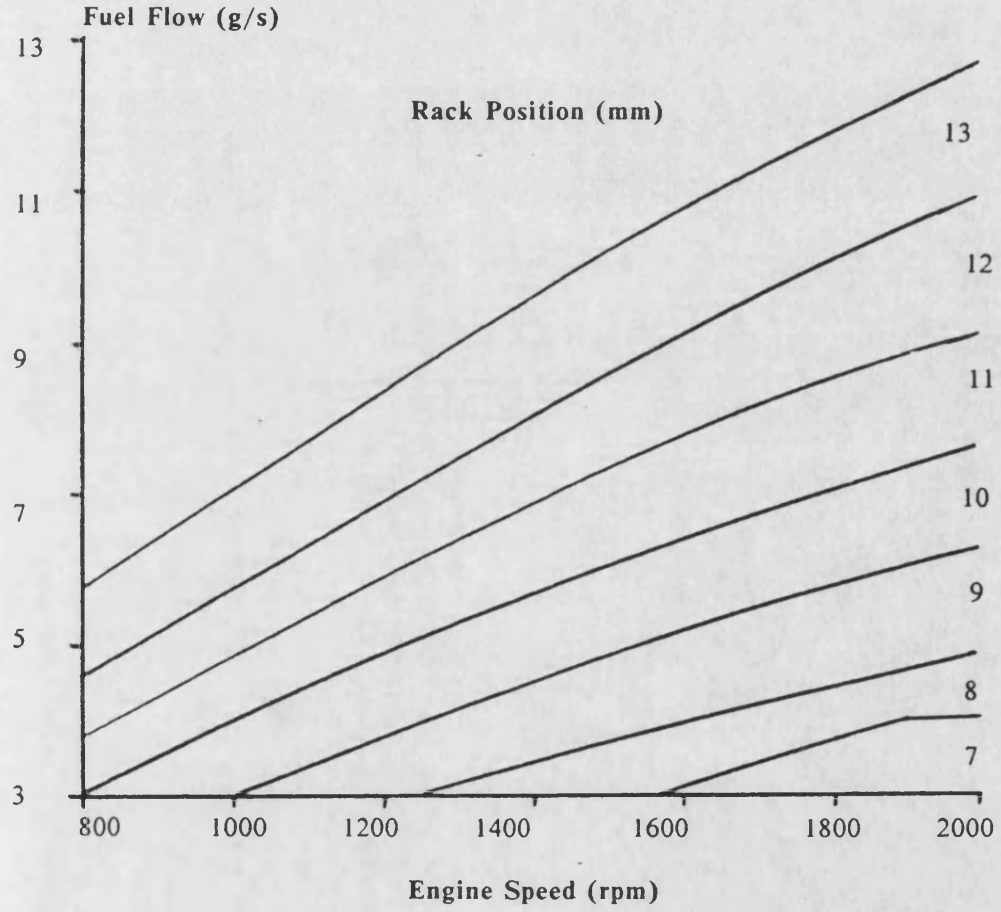


Figure 2.8 **The steady state performance characteristic for the fuel delivery system.**



CHAPTER 3.

THE BATH UNIVERSITY PARALLEL COMPUTER (B.U.P.C.).

3.1. A description of the computer hardware.

The University of Bath, School of Electrical Engineering parallel computer [23] [24] is based on the Motorola MC680XX family of microprocessors. It is implemented on a sixteen bit V.M.E. like back-plane with enhancements to give improved back-plane performance. The computer consists of a number of single board computers which each have their own local memory. This memory is dual ported to occupy a unique memory segment within the address space of the MC68000 processor on the multi-processor bus of the computer.

The architecture of the computer may be described as distributed shared memory. The use of the distributed memory avoids the processor execution 'bottleneck' caused by a true shared memory architecture. A true shared memory architecture provides a common memory area which all processors may access through the back-plane. A diagram illustrating the distributed shared memory architecture is given in figure 3.1.

A distributed shared memory architecture allows fast access between processors because of its tightly coupled nature. Communicating processors may read data from other processors directly. By comparison a loosely coupled system has to communicate all data specifically to the processors that need it. This is slower but is more secure. A tightly coupled system is less tolerant to individual processor failure.

For the research described in this thesis the speed of computation and communication is considered to be the most important factor in the choice of computer hardware. The BUPC is shown in figure 3.2.

3.1.1. The MC68000 single board computer.

The MC68000 board [23] (figure 3.3) is a general purpose processing board. It consists of a 12.5 MHz MC68000 [25] and one megabyte of dual ported dynamic memory. The board has a direct memory access (D.M.A.) controller, the HD68450 [26], which allows fast memory transfer and transparent disc memory transfers. Two memory management units, the MC68451, allow the use of the computer in a multi-user, multi-processor computer. The board has a S.A.S.I. parallel interface implemented using the MC68230 parallel interface timer. This gives an interface to Winchester hard discs and tape streamers. A Western Digital WD279X-02 disc controller provides an interface for up to four floppy discs. The board provides two serial RS232 interfaces, using the Syntek S6551, for connecting to terminals and printers. The board has no hardware floating point capability and so all floating point calculation on this board is implemented in software.

The MC68000 board is used as a 'master' processor in the computing system and provides all of the input and output for the computer.

3.1.2. The MC68020 single board 'slave' processing board.

The MC68020 board [24] (figure 3.4) is a dedicated processing board. It comprises a 16.667 MHz MC68020 [27] thirty-two bit processor with a MC68881 [28] hardware floating point co-processor. The co-processor implements the full I.E.E.E. standard [29] for binary floating point arithmetic. The co-processor also provides trigonometric and transcendental functions. The co-processor is fully integrated with the MC68020 and acts as an extension to the MC68020 register and instruction set. As with the MC68000 board, the MC68020 board has one megabyte of dual ported dynamic memory to the processor and to the computer

back-plane. The MC68020 has an enhanced instruction set over the MC68000 and this allows certain operations to be achieved with greater efficiency. The MC68020 processor implements a three stage instruction pipeline which allows the simultaneous execution of processor instruction, next instruction address decode and bus access. The processor has a sixty four word instruction cache which allows enhanced speed when executing short programming loops. The use of the more powerful processor with its hardware floating point capability gives this processing board a five to ten times speed up over the MC68000 board dependent on the amount of floating point calculation.

There is no input or output on the MC68020 board apart from two RS232 serial interfaces and an eight bit programmable L.E.D. display. Both these interfaces are provided using the MC68681 dual universal asynchronous receiver-transmitter. The MC68681 also provides the programmable timer for the MC68020 board.

In the B.U.P.C. the MC68020 board is used as a 'slave' processor. All of the loading of programs from disc and the initiation of programs is provided by the 'master' MC68000 processing board.

3.1.3. The shared memory back-plane.

The shared memory back-plane of the BUPC was originally designed for a parallel computer using the MC68000 single board computer for both its 'master' and 'slave' processor boards. It therefore uses the MC68000 memory map of sixteen megabytes and has a sixteen bit data bus.

Theoretically a maximum of sixteen one megabyte processor boards may be mapped into the MC68000 address space. However in the B.U.P.C. all the single

board computers read their local memory from address zero upwards. The bottom one megabyte of the address space cannot therefore be mapped to any particular processor. This mapping allows uniformity in programming of 'slave' processors without the need to account for the absolute address' of each board. Another one megabyte of address space is taken as an input/output area into which special back-plane cards, which provide additional computer functions, can be mapped. This leaves the possibility of up to fourteen processing board, including the 'master' board, in the computer.

Memory accesses between processor boards are controlled jointly by the two processor boards and the computer back-plane arbitration card. The requesting processor first obtains the computer back-plane by requesting the system arbiter card. When it has obtained the back-plane it interrupts the remote processor for access to its local bus. This interrupt routine is executed at the highest priority of the local processor as the requesting processor holds the back-plane during the operation preventing other processors from using it. When the remote processor allows access onto its bus the requesting processor executes its instruction and then releases both the remote processor bus and the back-plane.

3.1.4. The back-plane arbiter card.

The back-plane arbiter card controls all accesses by the processing boards of the computer to the back-plane. It implements a four level fixed arbitration scheme with daisy chaining of more than one processor on to each arbitration level. The arbitration level of each board is switch selectable. The position of the computer board in the rack with respect to the arbiter determines its priority within each arbitration level daisy chain.

3.1.5. Support hardware.

In addition to the main processing power of the computer, specialist cards are used to provide extra facilities. These are memory mapped into the input-output memory area. The specialist cards are useable by only the 'master' MC68000 board directly. The 'slave' MC68020 boards do not provide the necessary input-output interrupt circuitry to use the specialist cards directly.

3.1.5.1. The EFCIS graphics card.

The EFCIS graphics card provides eight colour graphics output. It gives two 512 by 256 pixel display pages which allows smooth animation. Pictures are drawn on a background page before being switched to the foreground to be displayed. The card is based on the Thompson EFCIS EF3965 graphics controller chip. The graphics card is used for both animated pictures and as a static output for the graphics kernel system (G.K.S.) [30].

3.1.5.2. The Multilink local area network card.

The multilink [31] card provides local area network facilities. Multilink is a cheap local area network allowing data transfer between computers and remote terminal access of the computer. It also allows the sharing between computers of printing and plotting facilities.

3.1.5.3. The global back-plane memory card.

The quarter megabyte global memory card provides an area of memory which all processor boards may use for the storage of variables. The global memory is used by multi-processor TRIPOS and this is described in section 3.2.

3.2. A description of the computer software.

3.2.1. The TRIPOS operating system.

TRIPOS [32] is a compact, single processor, multi-tasking operating system developed originally at Cambridge University. It is based around an assembly coded kernel and device drivers. The remainder of the operating system is coded in the high level language BCPL. TRIPOS provides a number of standard tasks for file and console handling. The fundamental devices are for a timer, a terminal and a disc.

The fundamental method of communication and control in TRIPOS is message passing. Messages are passed as packets of information. A packet is a pointer to a block of information containing an action, a destination, a link and additional specific data. The link allows packets to be joined to TRIPOS task and device work queues which are in the form of linked lists. Packets are controlled by the two fundamental communication primitives QPKT and TASKWAIT. QPKT links a packet onto the work queue of the destination task. A task uses TASKWAIT to unlink the head packet from its work queue or if the queue is empty to wait for another packet to arrive in its work queue.

TRIPOS tasks have fixed priorities and may be in one of three states: running, runnable or suspended. The highest priority runnable task is executed. Running tasks may suspend using TASKWAIT. Suspended tasks, of a higher priority than the running task, may be made runnable by other tasks. This occurs if the suspended task is sent by another task a packet using QPKT. If a task of a higher priority to the task which is running is made runnable then the executing task is suspended and the new highest priority task is executed.

TRIPOS is a single user operating system and does not have the memory protection of operating systems such as UNIX. Therefore it gives a very low operating system overhead of less than 1%. A comparable figure for UNIX on the MC68000 is for an overhead of 50%.

3.2.2. Multi-processor TRIPOS overview.

Because of its multi-tasking, TRIPOS is ideal for transferring to a multi-processor computer. It is small, compact and largely written in a high level language BCPL. At the time of its implementation, TRIPOS had already been developed for the MC68000 and the sources for the operating system were available for modification [33] [34] [35].

Multi-processor TRIPOS [23] is based on the idea of having individual 'slave' TRIPOS operating systems on every processor board. Tasks on each processor board are able to communicate with tasks local to the 'slave' TRIPOS and with tasks on other 'slave' TRIPOSes.

3.2.2.1. The 'Master' processor TRIPOS.

The TRIPOS on the 'master' processing board is similar to that for a single processor TRIPOS. It has all the file and console handling tasks as well as all the standard devices. To these is added an IOSERVER task which supports the 'slave' processing board TRIPOSes. The 'master' TRIPOS provides the user interface for the computer.

The IOSERVER is responsible for the management of the multi-processor TRIPOS global memory. All 'slave' processors use this task to obtain global memory. A simple form of output from the 'slave' TRIPOS is provided by the 'master' processor IOSERVER task using the console handler task. This facility is useful as an aid in debugging programs on the 'slave' TRIPOS.

3.2.2.2. The 'Slave' processor TRIPOS.

The 'slave' TRIPOSes are smaller than a standard TRIPOS operating system. They require no console and file handling tasks as this has to be handled by the 'master' processor. Each basic 'slave' TRIPOS consists of a REMEXEC task and a 'slave' debug task. Only a clock driver is needed in 'slave' TRIPOS.

The REMEXEC task allows other processors to access the 'slave' TRIPOS's internal functions. These include the task and device loading routines, the memory allocation routines and task management routines. This allows the loading of tasks and devices on to the 'slave' TRIPOS from the 'master'. A full description of these is given by Dale [23].

The 'slave' TRIPOS debug task operates in conjunction with the 'master' processor TRIPOS debug task and gives the user access to similar debug facilities for the 'slave' processors as are available for the 'master' processor. These are also outlined by Dale [23] and Berry [24].

3.2.2.3. Multi-processor TRIPOS inter-task communication.

Tasks on different processors can communicate directly in multi-processor TRIPOS. To achieve this, each processor is allocated a work queue pointer in global back-plane memory. Packets sent to the respective processors are added to the list of packets in the work queue pointed to by this pointer. As the computer is multi-processor, these lists must be protected to avoid concurrent writing of the lists and hence list corruption. This is achieved using a list access flag for each processor work queue which is also located in global back-plane memory. This flag is tested using the indivisible test and set instruction (TAS) of the MC680XX to obtain access to the queue pointer. Processors denied access to the list are made to busy wait on the flag, polling the flag until they do gain access.

If, on attaching the packet to the queue, it is found to be empty, then the sending processor will interrupt the receiving processor. The receiving processor will then execute an interrupt routine which deals with the whole of its work queue.

To distinguish between tasks on different processing boards all tasks in the computer system are given unique identifiers. The most significant bits of the task number reflect the memory slot of the processor board, whilst the lower bits reflect the local processor task.

3.2.3. Operating system 'boot' and computer startup.

The 'master' processor TRIPOS is started with a 'boot' program in PROM. This loads the complete 'master' TRIPOS image into memory and starts it. The 'master' TRIPOS loads the 'slave' TRIPOS images on to the 'slave' processors. It starts the 'slave' operating system by halting the 'slave' processor, loading the start address of TRIPOS into the reset location of the 'slave' processor, resetting the processor and releasing the processor.

A full description of the implementation of multi-processor TRIPOS onto the original MC68000 computer is given by Dale [23]. The modifications made to this to implement it on the MC68020 computer are discussed by Berry [24].

3.2.4. The BCPL programming language.

BCPL [36] [37] (Basic Combined Programming Language) is a block structured language. It was developed by Martin Richards in 1967 and is based on CPL (Combined Programming Language) which was never fully implemented. It has been designed as a tool for writing compilers and systems development. The operating system TRIPOS is written using BCPL.

BCPL allows separate compilation and introduces the concept of a GLOBAL data area to allow this. It is a typeless language having only one data type, the machine binary word. Several assembler type operators such as shift, AND, OR, and NOT are implemented. This gives the advantages of assembler speed with the readability of higher level code. BCPL also gives the programmer access to the address values of variables. This is particularly useful in the reading of variables on remote processors in a shared memory architecture. However, it has to be

used with care as it is inherently 'unsafe'. BCPL provides a full set of high level programming constructs and allows the use of recursion.

The BCPL compiler produces code that is compact and that compares favourably with that for hand coding. Where hand coding is necessary, for example where the indivisible 'Test and Set' instruction of the MC680XX processors is needed, hand coded routines can easily be incorporated using the GLOBAL data area. These assembly coded routines can be separately assembled and joined to the main program.

The BCPL compiler used at Bath University has been extended to incorporate floating point operations. These are typeless as the results from such operations are still binary words. The programmer is totally responsible for ensuring that results from these operations are used correctly and that the values operated on are appropriate to floating point operations.

Further work has been carried out on the BCPL compiler to increase the efficiency of floating point operations when using the MC68020 board with its floating point co-processor [28]. Most of the trigonometric and transcendental functions have been made direct BCPL operators rather than library procedure calls. This speeds the execution of these functions and is particularly useful in the area of simulation where speed of floating point operation is critical.

The BCPL compiler has been improved to make full use of the MC68881 co-processor. In the original implementation only one of the floating point registers had been used. This meant that all floating point operations were individually loaded onto the co-processor. By using the full set of floating point registers, intermediate floating point calculation values can be retained in the co-processor

rather than being continually rewritten in and out of the co-processor. This work on the compiler gave a 50% speed up in simulation speed.

BCPL gives the programmer a great deal of freedom in program structure. However it does not give the compilation error checking of high level languages such as Pascal or 'C'. It is ideal for applications where computation speed is important. BCPL was used in all the coding of the multi-processor diesel engine simulator in this research.

3.2.5. Program support and debugging facilities.

3.2.5.1. Remote task handling functions.

The set of TRIPOS commands has been extended to give specialist multi-processor commands. These allow the loading of tasks, devices and libraries onto remote processors as well as the control of these tasks. A full outline is given by Berry [24].

3.2.5.2. The debug task.

The original TRIPOS debug task has been expanded to allow the monitoring of both the 'master' and 'slave' processor TRIPOSes. The user may select any of the processors and examine its memory, obtain disassembly and look at the processor registers. A full outline of the debug task is given by Berry [24].

3.2.5.3. The MC68020 processor board utilisation display and the back-plane access monitoring LED.

A task has been developed to monitor 'slave' processor utilisation. The utilisation is calculated by monitoring the time spent by each TRIPOS, on each processor, in its idle task. The idle task is the task run by TRIPOS when no other is runnable. The idle task is made to execute a loop count. This count is monitored by a processor utilisation task which converts this value into a processor time usage figure. This can be monitored by the user from the 'master' processor.

The MC68020 processor board has an eight LED display panel. This is used as a simple visual display of processor usage. Such a display allows rough approximations of task distribution to be made. This allows the user to determine the most highly loaded tasks. The user can then correct this to allow a more evenly distributed problem where possible. This can reduce the chance of heavily loaded processors becoming 'bottlenecks'.

Another LED display on the processor board indicates processor back-plane accesses. Intensity of the LED give an impression of back-plane usage.

3.3. The limitations of the parallel computer.

Theoretically up to fourteen processors can be placed in the B.U.P.C. However experiments have determined that this maximum number is unattainable. This is due to processor memory failure when this number is used. A test was carried out in which all processor boards accessed each others memory repetitively. When fourteen processors were used processor failure was observed. The reasons for this and a general critique of the B.U.P.C is given in the following sections.

3.3.1. Fixed priority scheme back-plane arbitration.

Processors whose accesses to the back-plane conflict are allocated use of the back-plane on the basis of their set priority and daisy chain level. When a high priority processor requests the back-plane at a high enough rate, a lower priority processor can be locked off the back-plane. This could never occur in the original MC68000 system as all processor instruction fetches took at least one local memory read which had the effect of always releasing the back-plane. However with the MC68020 system, the use of the cache and the instruction pipeline allows a processor to access the back-plane continually. This is an unsatisfactory situation as care has to be taken in the allocation of processing tasks to processors to avoid heavy use of the back-plane by higher priority processors. For an ideal parallel computer, this would not be the case and ideally the parallel programmer should not have to fit his task allocation to suit a particular computer.

Multi-processor TRIPOS communication involves busy waiting. This occurs when access is denied to the work queue of another processor by that queue's protective flag. Care has to be taken as the high priority processors can block the release of a packet queue by a low priority processor by continually accessing the queue's semaphore. This causes a 'deadly embrace'. The problem has been partially fixed by adding a delay into the busy wait loop. The delay is determined by the number of processors in the computer. The delay is sufficient to allow all processors to access the bus between flag tests. It should be noted that this only takes account of this particular case of bus usage and other routines with heavy bus usage may effect it and still cause the problem. The delay has the effect of slowing down the speed of inter-processor communication. Similar delays have to be incorporated into specialist synchronisation routines

which involve busy waiting on remote memory locations.

3.3.2. Back-plane access and its effect on local memory refresh.

The mechanism for requesting use of the back-plane by the processor boards means that a processor that is barred entry to the back-plane by the arbiter card is suspended in mid-processor cycle. This has the effect of stopping all local processor bus actions including the dynamic RAM memory refresh. If a processor is held off the back-plane for long enough by a higher priority processor then the processor will have its memory corrupted by refresh errors and will crash. With the original MC68000 system this did not occur as the back-plane usage could not become saturated.

In the implementation of the diesel engine model described in the next section, great care has had to be taken in the allocation of processing tasks to processors. This is to ensure that memory corruption does not occur. This has the effect of limiting the number of useable processor 'slave' boards to eleven rather than twelve. However with eleven processors, it is still possible to have problems and only certain permutations of task allocation will work.

It should be noted that many permutations that failed did so after some time and some indication as to their characteristics could be obtained. The permutations that failed most swiftly were those that had a higher back-plane usage. This is shown both by a quicker simulation speed and from an analysis of back-plane usage which is discussed in the next section.

An analysis has been made of back-plane bus utilisation. By monitoring the voltage of the computer back-plane bus grant acknowledge line, the amount of utilisation can be obtained. This is achieved by comparing the measured voltage with the voltages for no bus access and maximum bus access. For the parallel diesel engine model, based on control volumes, produced by this research, a value of 56% bus utilisation has been measured. This indicates a one in two chance of bus contention and explains the difficulty of using an increased number of processors and certain permutations of tasks on the computer.

The ability of the MC68020 board to suspend lower priority processing boards in mid-cycle can be modified by making the MC68020 have a bus grant time out. On time out the processor can execute a hardware retry thus allowing the refresh circuitry to obtain the processor bus.

3.3.3. Arbitration daisy chain.

For each of the fixed arbitration levels of the B.U.P.C. a daisy chain of processors can be attached. The length of this daisy chain is limited by the physical limits of the logic used in the implementation of the daisy chain. The arbitration card produces a back-plane grant signal of a finite length. This is passed down the daisy chain until a processor that has requested the use of the bus is reached. This processor then accesses the bus. All the MC68020 processor boards have asynchronous clocks. This has the effect of reducing the length of the bus grant signal as it traverses the chain due to logic delays and the difference between adjoining processor clocks. This limits the possible number of MC68020 processors in the computer to a daisy chain of three MC68020 processors. The effect is not observed for the 'master' MC68000 board as its clock is synchronous with that of the arbiter card and the back-plane. The

number of 'slave' processing boards is therefore limited to twelve for the MC68020 system.

3.3.4. Back-plane communication bandwidth.

The MC68020 is a thirty-two bit processor and therefore a sixteen bit back-plane is a serious restriction to inter-processor communication. BCPL, the language used in this research, has only one data type, that of the word. For the B.U.P.C. the word length is thirty-two bits. Thus all inter-processor actions take two back-plane cycles.

An upgraded system should use a thirty-two bit back-plane thus allowing increased communication speed between MC68020 boards. As communication is thirty-two bit it will also reduce the number of accesses needed by individual processors by two and therefore reduce the amount of inter-processor contention for the back-plane.

3.3.5. Limited maximum number of processors.

The MC68000 memory map allows a maximum of sixteen one megabyte processors in the computer system. The use of the MC68020 memory address map would allow an increase in the possible number of processors in the computer from fourteen to 4094 or an increase in the size of each processing board's memory. An increased possible number of 'slave' processing boards would allow the investigation of 'finer grain' parallel problems. The present limited number of processors precludes such investigation. This increased number of processors will also require the implementation of a new arbitration scheme. An increase in the memory size of the processing boards would allow increased space for program

data structures and data logging.

3.4. Future developments.

Using the present parallel computer based on the MC68020 a high speed diesel engine simulation has been developed. The increased speed of the simulation has indicated new areas of application for such a model. These application, which are discussed in chapter 9, would benefit from even higher simulation speed. A reasonable speed improvement would enable the simulation of the diesel engine model in 'real time'. It is envisaged that this increased speed will be obtained both through increased parallelism and more powerful computing hardware.

3.4.1. Bath University Transputer system.

A new parallel computer is at present being developed in the School of Electrical Engineering [38] [39]. This will be based on the floating point transputer, the Inmos T800. The transputer has been designed for parallel processing. It has four serial communication channels or 'transputer links' which may be linked with other transputers to form any network topology.

The speed of the serial links is nominally 10 Mbaud. From an analysis of the parallelism of the diesel engine [40], it has been determined that this communication rate is too slow. Therefore a shared memory transputer system is being designed, based on a new overlapped cycle multi-processor bus. This will give high speed communication between processors. The computer will consist of transputer clusters connected together via 125 Mbaud optic fibre channels. Each cluster will consist of sixteen transputer cards, a fibre optic card, and a universal 'transputer link' configuration card. The 'link' topology card will be

used to configure the transputer link topology and will in addition allow up to thirty two input/output links. The shared memory system is designed so that all transputers may write to one another at full speed. Thus there is no time penalty for inter-processor communication using writes. The system also allows the concept of 'broadcasting' where one transputer may write to all the transputers in the system at the same time. The system is designed to allow a maximum number of 256 clusters or a total of 4096 transputers. A brief specification is given in appendix B.

3.4.2. The Helios operating system.

The new computer system will use a commercial operating system known as Helios [41]. Helios is designed for multi-processor systems of any configuration or size. It is a distributed operating system with the multi-processor nature of the system hidden from the user and programs. The programming language for the new system will be ANSI 'C' [42] because the operating system is written in this language and it is a widely supported commercial language with a recognised international standard.

The use of a widely used language on a commercially produced operating system will increase the portability of the parallel diesel engine simulation.

Figure 3.1 The distributed shared memory architecture.

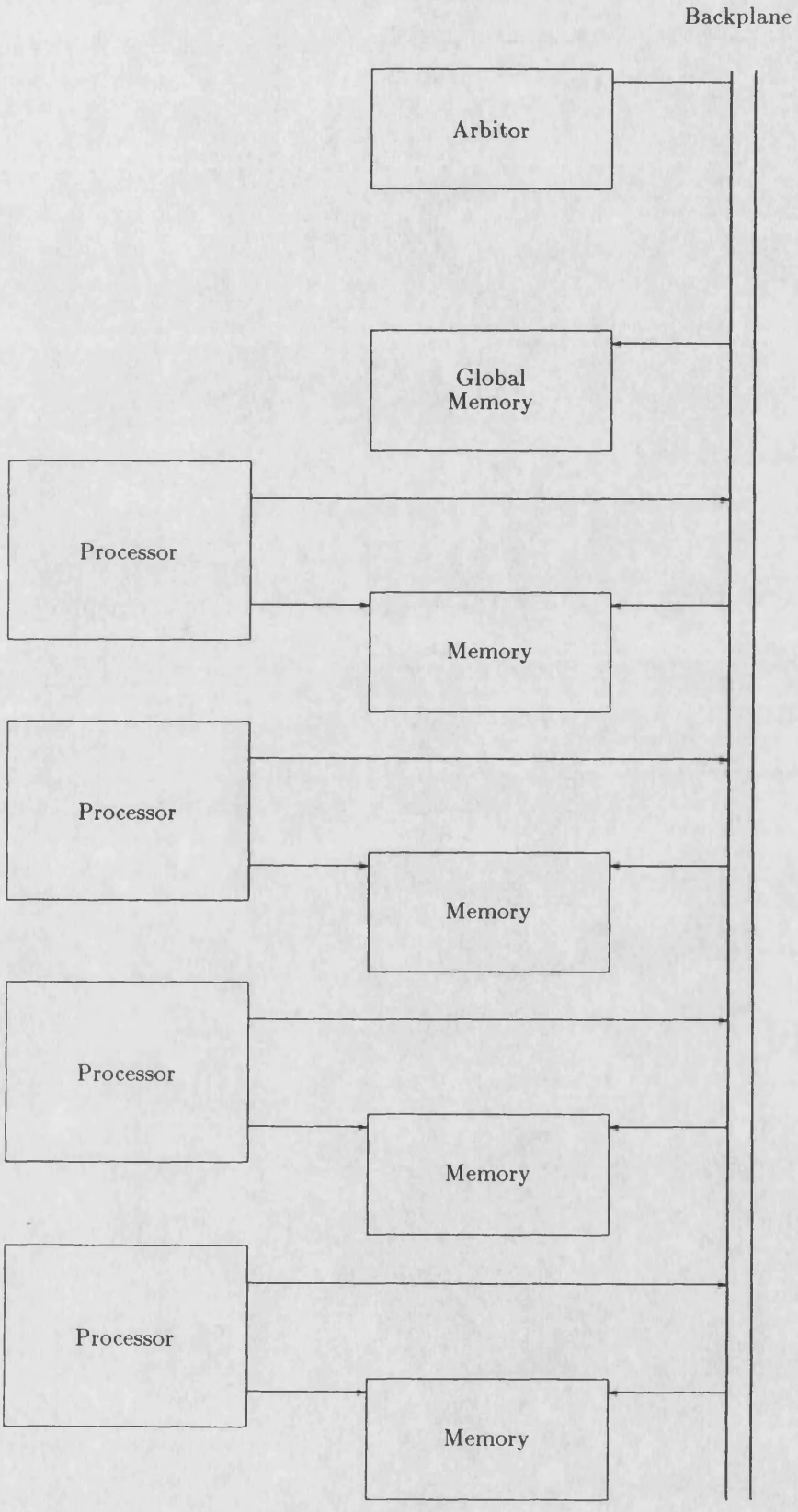


Figure 3.2 The Bath University Parallel Computer.

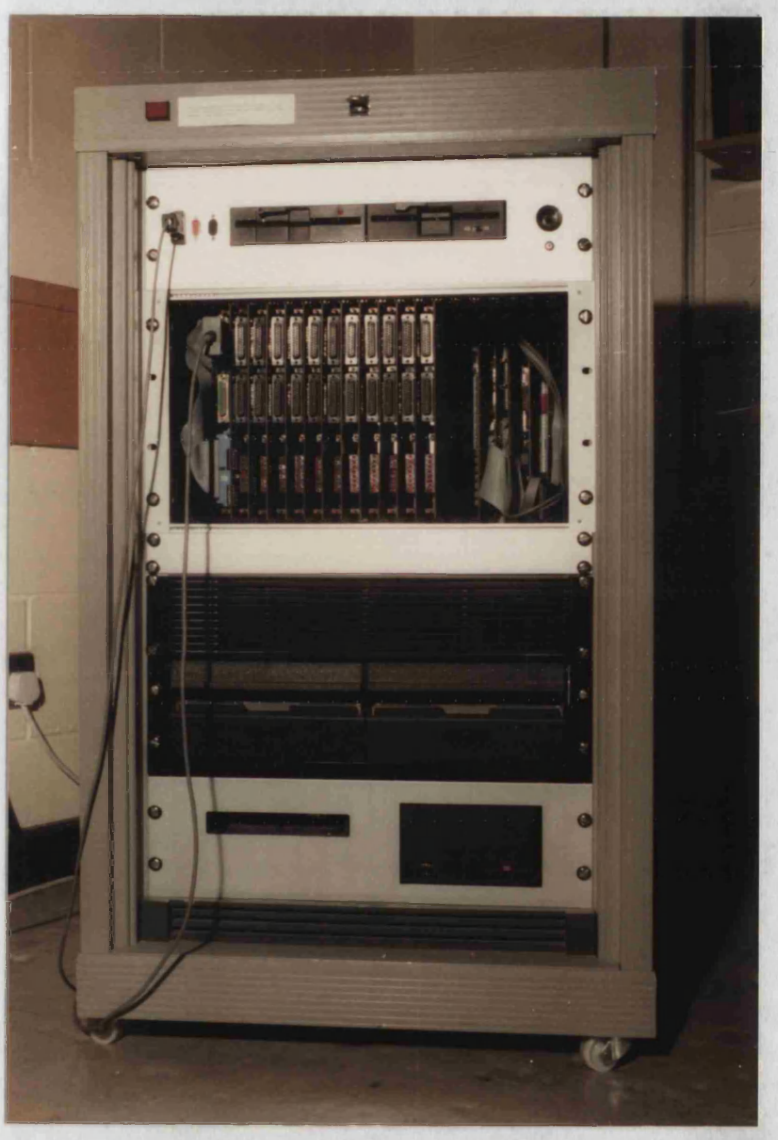


Figure 3.3 The MC68000 single board computer board.

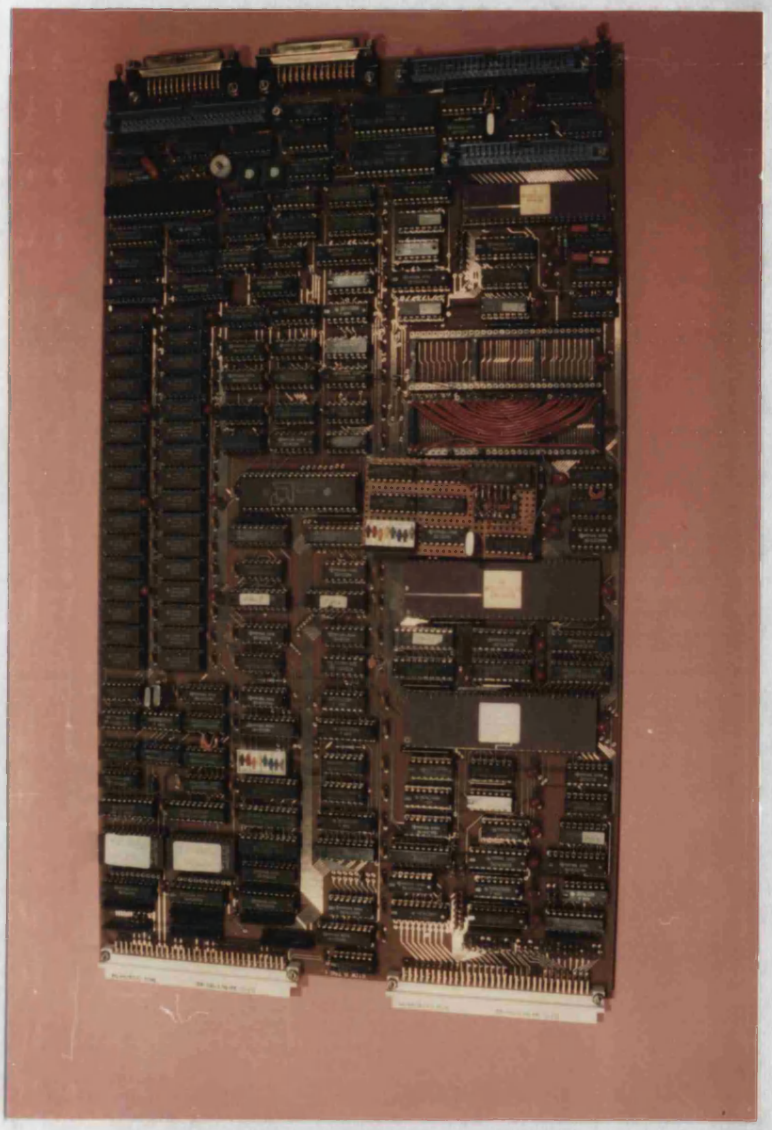
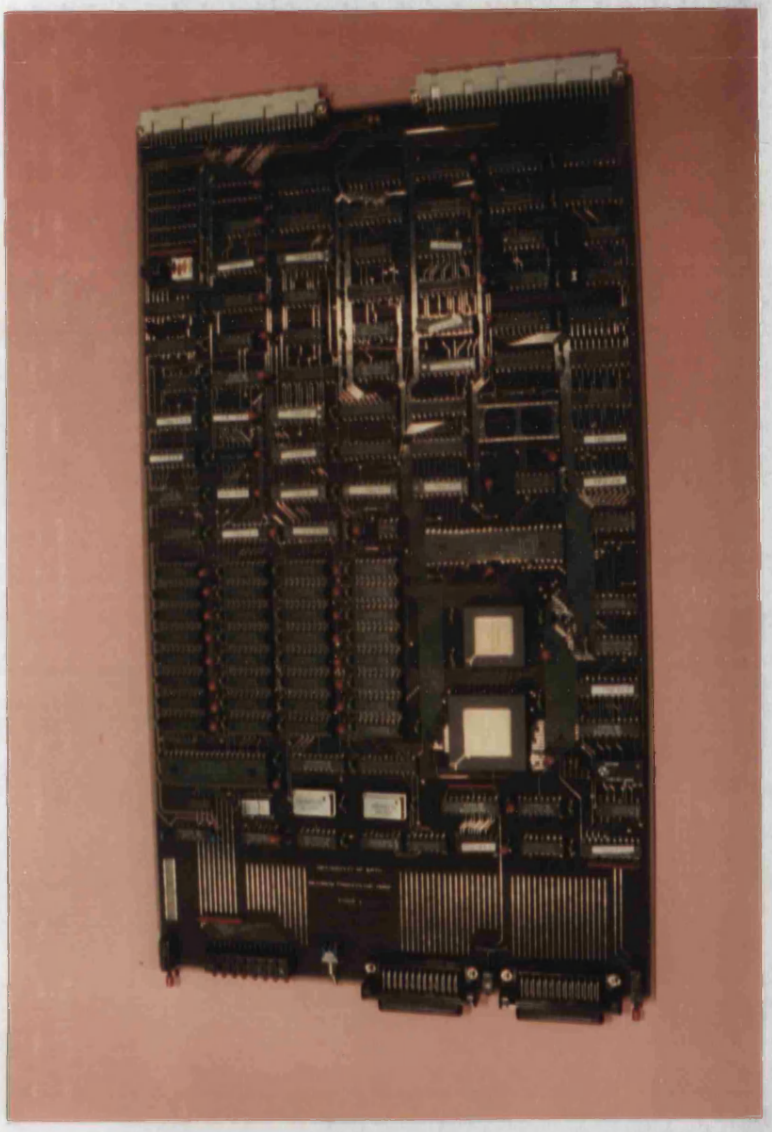


Figure 3.4 The MC68020 'slave' processing board.



CHAPTER 4.

THE PARALLEL SIMULATION OF A MULTI-VOLUME DIESEL ENGINE.

4.1. Introduction.

This chapter discusses the parallel nature of the 'filling and emptying' diesel engine simulation. The first section describes the two previous parallel diesel engine simulations produced by Jones [8], with a detailed criticism of the parallel methods. From these criticisms the design criteria for a new parallel diesel engine simulation are described and the practical implementation of this new parallel simulation is given. The chapter concludes with a comparison of simulation performance for the new and old simulation.

4.1.1. Parallelism in multi-volume diesel engine simulations.

As described in chapter two, the 'filling and emptying' method views a diesel engine as a set of interconnected thermodynamic control volumes. Superimposed on this are dynamics functions and control functions which describe the actions of moving parts within the engine and the actions of the engine control systems. The state of each control volume in the diesel engine can be determined from three ordinary differential equations. These represent the control volume temperature, fuel-air ratio and mass.

The 'filling and emptying' method when applied to a multi-volume engine exhibits inherent parallelism. The calculation of each control volume state in the engine simulation is largely independent from the calculation of other control volume states. Additional parallelism is present in the calculation of the dynamics and control functions of the engine. These have well defined communication with the rest of the system and may be calculated concurrently to the control volume calculations. This parallelism, based on the control volumes of the engine, may be termed 'geometric'. 'Geometric' parallelism is where the degree and type of

parallelism is dependent on the structure of what is being modelled.

4.1.2. The structure of the parallel engine simulations under TRIPOS.

In all the parallel diesel engine simulations described in this chapter the parallel sections have been coded as TRIPOS tasks for the Bath University parallel computer. Each of these tasks may be executed on different processors in the computer. TRIPOS tasks are covered in more detail in chapter 3. The coding of the simulation into TRIPOS tasks allows the easy implementation of the simulation onto various numbers of processors. It also allows the use of the TRIPOS communication primitives for data transfer between parallel sections and for the control of the parallel sections.

4.1.3. The Leyland TL11 engine simulation structure.

The multi-volume diesel engine simulation described in this thesis is of an eleven litre Leyland TL11 truck engine. A fully instrumented TL11 engine, in the school of mechanical engineering, has been used to validate the parallel diesel engine simulation. The fully instrumented TL11 engine is described in chapter 7.

The TL11 engine consists of six cylinders, two exhaust manifolds and one inlet manifold. The manifolds are all connected to a variable geometry turbo-compressor. The TL11 engine has been modelled as six cylinder control volume tasks, two exhaust manifold volume tasks, one inlet manifold volume task, a dynamics task and an engine actuator control task. The calculation of the flow through the compressor is carried out by the inlet manifold task. The flows through each half of the turbine are calculated by each of the exhaust manifold volume tasks. The simulation structure for the TL11 diesel engine is given in

figure 4.1. This division, into parallel tasks, has been used in all of the parallel TL11 engine simulations described in the following sections.

4.2. The integration method used in the parallel diesel engine simulations.

The 'filling and emptying' method consists of three ordinary differential equations for each engine control volume. In all the simulations outlined in this chapter, the differential equations have been solved using the modified Euler predictor corrector method. This is arguably the simplest of the predictor corrector algorithms. A predictor corrector algorithm uses both an explicit and an implicit integration formula to solve an ordinary differential equation. Each newly calculated point is first predicted using the explicit formula and is then corrected using the implicit formula. The equations for the modified Euler predictor corrector algorithm are given in equations 4.1a and 4.1b.

$$y^p_{n+1} = y^c_n + h f^c_n \qquad \text{Equation 4.1a}$$

$$y^c_{n+1} = y^c_n + \frac{h}{2}\left(f^c_n + f^p_{n+1}\right) \qquad \text{Equation 4.1b}$$

A diagram showing the application of the modified Euler predictor corrector method is given in figure 4.2.

Annand [43] discusses the choice of integration formulae for engine simulation. He compares the use of Runge-Kutta methods with predictor corrector methods. Using a fourth order Runge-Kutta integration method, he found that incorrect solution was given in the region of valve overlap or scavenge. This problem was not observed with predictor corrector methods. Annand also notes a higher speed of calculation with a predictor corrector algorithm.

Although the modified Euler integration method is the simplest of the predictor corrector formulae, it is considered to be sufficiently powerful for diesel engine simulation. This is because the engine volume equations are not generally stiff. A problem that is said to be stiff is one in which the solution components of interest are slowly varying but where solutions with very rapidly changing components are possible [44]. The diesel engine simulation equations are stiffest during valve overlap or scavenge in the cylinder control volume. This is due to small gas flows through both cylinder valves and to a very small cylinder volume.

The solution of the control volume equations proceeds with respect to engine crankshaft angle. For each calculation step the state of each control volume, temperature, fuel-air ratio and mass, is predicted and then iteratively corrected using the modified Euler equations. The iteration of the corrector formula is continued until stability is reached on all the engine control volumes. The stability criterion used is given in equation 4.2.

$$1-\epsilon < \left(\frac{y_c}{y_p}\right) < 1+\epsilon \qquad \text{Equation 4.2}$$

The stability criterion is based on a comparison of the previous two iteration results. Stability must be reached across all the system control volumes as the volumes are interconnected and therefore the state of each affects the state of the others.

For some calculation steps integration stability cannot be reached. This will occur where the simulation equations exhibit stiffness. The result of this will be an oscillating integration solution outside the bounds of integration stability. The modified Euler predictor corrector algorithm allows the reduction of the

integration step length and this increases the stability of the integration method by reducing the integration error. Integration step length reduction can be used when stability cannot be reached at a particular step length. In the simulations described in this chapter, an upper limit of three is applied to the number of iterations of the corrector formula. When this upper limit is reached the integration step length is halved.

4.3. Historical development of parallel diesel engine modelling.

4.3.1. SPICE.

The engine simulation program SPICE was developed by Charlton [16]. It is a large general purpose simulation program coded in Fortran. The program SPICE is based on the 'filling and emptying' method and allows a user to simulate a wide range of engine configurations. SPICE was designed for use on serial mini-computers. The program SPICE was used as the basis for the parallel diesel engine simulations developed by Jones [8].

4.3.2. Parallel diesel engine simulations.

Jones [8] developed two distinct types of parallel diesel engine simulation. These simulations were based on the control volume parallelism discussed in section 4.2. The first type of simulation was based on communication between the engine control volume tasks at each solution step. This simulation had a centralised control structure and is described as the centralised solution step communication (CSSC) simulation. The CSSC type of simulation is described in section 4.3.3. The second type of simulation was based on communication between engine control volume tasks at each integration iteration step. This simulation also has a

centralised control structure and is described as the centralised iteration step communication (CISC) simulation. The CISC type of simulation is described in section 4.3.4.

4.3.3. The simulation of a multi-volume diesel engine with communication between parallel tasks at every solution step. (CSSC simulation structure)

In the CSSC simulation, each of the engine control volumes is viewed as independent during each integration calculation step. Calculation in each control volume is based on the state of any connected control volumes and the state of the dynamics tasks at the end of the previous calculation step. By separating the control volumes in this way the parallel sections of the simulation can clearly be defined and the communication between them can be restricted. Iteration of the predictor corrector formulae for each control volume and possible integration step length reduction is controlled by the control volume task. Because information from the other engine control volumes is assumed constant during each calculation step the iteration of the integration method does not need to be synchronised across the whole system.

4.3.3.1. The centralised synchronisation of the simulation and the inter-task communication.

The CSSC engine simulation is controlled by a central synchronisation task. This task ensures that all simulation tasks are synchronised for every calculation step. At the beginning of each calculation step the synchronisation task starts the control volume, dynamic and actuator tasks by sending TRIPOS packets. The calculations for the step are then solved independently on each task until

equation integration stability is achieved. Each calculation task then synchronises for the next calculation step with the central synchronising task by returning the TRIPOS packets.

Communication of data between the tasks at the beginning of each calculation step is achieved in two ways:

- Data is communicated directly using predefined result areas.

- Data is passed through the central synchronisation task.

4.3.3.2. The problem of result inaccuracy due to control volume independence in the CSSC simulation.

The 'filling and emptying' method views the diesel engine as a set of interconnected volumes. The CSSC simulation has the effect of disconnecting each control volume from its neighbours, giving constant boundary conditions, for each calculation step. This could lead to anomalies such as the creation and destruction of mass in the valve flows between cylinders and manifolds. The cylinder volume task uses a constant manifold volume state for the calculation of the flow parameters through its valves. The manifold volume tasks use the constant valve flow parameters calculated by the cylinder volume task in the previous step for its step calculations.

An effect of this style of implementation is the increased ease in solution of each set of control volume equations. The use of constant boundary conditions, for each control volume on each calculation step, reduces the need for reiteration of the corrector formula and the possible use of step length

reduction. The control volume equations are effectively simplified and this gives a higher speed of computation.

4.3.4. The simulation of a multi-volume diesel engine with communication between parallel tasks at every iteration of the integration formula. (CISC simulation structure)

In the CISC parallel diesel engine the problem of the separation of the control volume solutions was corrected.

4.3.4.1. The centralised synchronisation task in the CISC simulation.

The CISC simulation is based around a central synchronisation task which is responsible for the synchronisation of all of the engine control volume tasks on each iteration of the predictor corrector formulae. In addition the central synchronisation task must also synchronise with all of the engine tasks for each integration solution step.

The synchronisation task coordinates the application of the predictor formula and the iteration of the corrector formula on all of the engine control volume tasks. All of the control volumes use the results from the previous iteration for the next calculation. The synchronisation task ensures that all the control volumes have achieved stability before the simulation proceeds to the next step calculation. When stability across the whole set of control volumes is not met within a user defined number of iterations, the synchronisation task will reduce the integration step length by half on all the control volumes tasks. This reduction in step length improves the stability characteristics of the integration solution.

4.3.4.2. The control volume tasks in the CISC simulation.

Each control volume task in the CISC simulation consists of calculation routines for the prediction and correction of the control volume state. These routines are passed the necessary start conditions for the calculation at the beginning of each iteration step by the TRIPOS packet from the synchronisation task. For example, the cylinder control volume is passed the engine speed, and the state of all of its connected manifold control volumes at the start of each calculation. At the end of the calculation, all of the control volume tasks return its newly calculated control volume state to the central synchronisation task.

4.3.4.3. Inter-task communication in the CISC simulation.

In the CISC simulation each control volume only communicates with the central synchronisation task. All communication between control volumes is therefore through the synchronisation task. The central synchronisation task uses the returned results from the calculation tasks in two ways. It directly sends them to the appropriate calculation tasks for the next calculation step. Alternatively it uses the results to calculate combined result values for the next calculation step.

A descriptive diagram of the CISC simulation is given in figure 4.3. It details the style of communication between parallel tasks and the control of the parallel tasks.

The parallel processing structure of the CISC simulation is of a processor 'farm' type. One central task 'the farmer' parcels out complete work packages to other processors 'the farmhands'. This style of parallelism in the diesel engine simulation is comparatively easy to program. This is because each of the control

volume tasks can be represented as a set of simple calculation functions with well defined inputs and outputs to just one central task.

4.4. A critical examination of both the CSSC and CISC parallel diesel engine simulations.

4.4.1. The use of centralised synchronisation.

In both the CSSC and the CISC simulations the operation of the simulation is based on a central synchronisation task. This central synchronisation task controls all aspects of calculation in all of the simulation tasks. Therefore, all other tasks in the parallel simulation are forced to wait for it. The effect of this is to produce a serial 'bottleneck' in the simulation where no parallel operation can occur.

For the research in this thesis, a CSSC type simulation was implemented on the MC68020 Bath University Parallel Computer. Using an allocation of one engine control volume to each processor in the computer the processor utilisation was analysed. This indicated that the central synchronisation task used 40% of the processing power of a single processor. This meant that all other tasks in the simulation were prevented from using roughly 40% of their processing power by having to wait for the central synchronisation task. For the CISC simulation the synchronisation task processor utilisation would be even higher as the central synchronising task handles all of the inter-task communication.

Central synchronisation of the diesel engine simulation forces all engine simulation tasks to wait for all other simulation tasks. This occurs whether or not the other tasks will have any effect on the waiting task. For example, the

cylinder has both of its valves closed for half of the engine cycle and is therefore unaffected by the manifolds during this period. For this half of the engine cycle, the cylinder control volume tasks only need to communicate with the engine dynamics task to obtain crankshaft speed.

It should be noted that both the CSSC and the CISC diesel engine simulations were developed on a MC68000 parallel computing system. The MC68000 processor has no hardware floating point capability and so all floating point calculation was emulated in software. Calculation time therefore was high because the diesel engine simulation equations are floating point calculation intensive. Inter-processor communication time for the MC68000 parallel computer is small compared to the processor calculation time. Inefficiencies in communication would not therefore have been a major problem when the CSSC and CISC simulations were initially implemented. In the MC68020 system used in the research for this thesis, communication inefficiencies cannot be ignored. Communication is still based on a sixteen bit computer backplane and the communication speed between processors in the computer is therefore the same. However the MC68020 with its floating point co-processor gives an increase in processing power of about ten times over the MC68000. Communication inefficiencies are therefore now of utmost importance.

4.4.2. The use of centralised data communication.

Two forms of data communication between simulation tasks are used in the parallel simulations. In the CSSC simulation some data is communicated directly, which is examined in the section 4.4.3. All of the data in the CISC simulation and some in the CSSC simulation is communicated through the central sychronisation task.

The communication of data through the central synchronisation task can take two forms:

- data is passed directly through the synchronisation task.

- data is processed by the synchronisation task prior to sending it out to the simulation tasks.

For example, the central synchronisation task processes the flow values between the cylinder and manifold control volume tasks. The synchronisation task is used to accumulate all of the valve flow values calculated by the cylinder control volumes. It combines these to form a single set of flow values for the manifold control volumes.

By using centralised communication the programming of a parallel simulation is made easier. However data transfer through the central synchronisation task is very inefficient. Data has to be passed twice rather than being read directly. As the synchronising task is the intermediate task in this data communication it also increases the time spent in synchronising. This increases the problem of the serial 'bottleneck' caused by the synchronisation task. The added processing of the communicated data, by the synchronisation task, increases the 'serial bottleneck' problem still further.

4.4.3. Non determinacy in direct task to task communication.

In the CSSC simulation most data communication between simulation tasks is achieved directly. This direct communication uses predefined result areas for each simulation task and all simulation tasks can directly read these result areas.

These result areas are setup in the initialisation of the parallel diesel engine simulation. Each simulation tasks updates its result areas at the end of every calculation step. The results are then read by other tasks during the next calculation step.

There is a major problem in such a simple communication scheme. The direct communication does not take account of the asynchronous nature of concurrent tasks. A task in the engine simulation can quite possibly complete its calculation and update its result area, before other tasks that should have read the previous result have done so. The reading task will therefore obtain the incorrect data for its next calculation or indeterminate data if the updating occurs concurrently with the reading. This effect became readily apparent with increased simulation speed due to more efficient program coding and the use of the faster MC68020 computer boards. The slower calculation speed of the MC68000 computer had masked this effect.

The non-determinacy in direct tasks communication was initially overcome using double buffering of the result areas. For this each communicating task was allocated two adjoining data areas. These data areas were used alternately for writing data or reading data. Each simulation task maintained an offset pointer to the read result areas on other tasks and to its write result area with these pointers being alternated for each calculation step. The two data areas were protected from overwriting by the synchronisation of the simulation with the central synchronisation task.

4.4.4. The non use of overlapped communication.

The two TRIPOS communication primitives QPKT and TASKWAIT take a finite amount of time to execute. This time can however be small compared to the time spent in calculation by a task, between receiving a TRIPOS packet and returning it to the sending task. This time can usefully be used for calculation by the task which has sent a packet. The sending task will otherwise suspend and wait for the other task. In both the CSSC and CISC simulations no use is made of this ability to overlap communication with calculation because central synchronisation is used. Two way communication, between tasks, occurs at fixed points in the calculation with the sending task suspending and waiting. Processor power is wasted waiting which can be used if a better communication structure is implemented. This concept of overlapping communication and calculation is shown in figure 4.4.

4.4.5. Inefficient program structure.

The cylinder control volume calculation equations vary with crankshaft angle. This variation is described in chapter 2. The cylinder calculation sectors occur in the following fixed order: scavenge, induction, compression, combustion, powerstroke, exhaust. In the CSSC and CISC simulations, the cylinder control volume tasks carry out a large set of crankshaft angle comparisons for each calculation step. These comparisons are to determine which cylinder calculation equations will be used. As the order in which the calculation sectors occurs does not change, it is far more efficient to only detect the next change in calculation equations and to change the calculation routine permanently at this point.

In both the CSSC and the CISC simulations each simulation task uses a large amount of intermediate calculation. This means that numbers are shuffled in and out of memory unnecessarily. By avoiding the use of intermediate variables a higher simulation speed can be obtained.

4.4.6. The problem of inaccuracy when using floating point operations for the control of the simulation.

In the diesel engine simulation, the crankshaft angle is used as an independent variable rather than time. In the manifold control volume tasks, the angle is simply used as a calculation steplength and no absolute knowledge of crankshaft angle is needed. In the cylinder control volumes the value of crankshaft angle is used by the cylinder control volume in the calculation of the following:

- cylinder surface area,
- cylinder volume,
- cylinder rate of change of volume,
- valve flow areas,
- the selection of cylinder calculation equations.

In the CSSC and CISC diesel engine simulations, the crankshaft angle is incremented at each calculation step until a full cylinder cycle of 720^{o} is complete. The crankshaft angle is then reset to zero.

Due to rounding off error in floating point calculation it was found that for a crankshaft angle step of 2^{o}, an extra step calculation was performed for each 720^{o} engine cycle. This would cause an error in the results from the simulation. The problem has been corrected by controlling the resetting of the crankshaft

angle to zero using an integer variable. The integer variable is incremented for each calculation step until the defined number of steps in the cylinder cycle is reached. An integer variable may be used for the resetting to zero when the calculation steplength is a divisor of 720^{o}.

The accuracy of the zeroing of the engine crankshaft angle is important as the calculation of engine cycle parameters such as engine Brake Mean Effective Pressure (BMEP) is determined by it. In a distributed control strategy, of the type outlined in section 4.6, it is important that cycle parameters are calculated for the same number of engine calculation steps on all control volume tasks. With distributed control of the engine simulation there is no central synchronising task to ensure this.

4.5. The design criteria used in the design of the new diesel engine simulation.

In the redesign of the parallel diesel engine simulation a number of design criteria were considered important. These are outlined in the following sections.

4.5.1. The need to avoid a centralised distribution of data in the parallel engine simulation.

To be efficient all data transfer should be direct between calculating tasks. This leads to a reduction in the amount of inter-task data communication. Direct communication is felt to be the only solution that can be usable in future parallel simulation work, when increased parallelism is identified and used. It should be noted that direct communication of data, between simulation tasks on different processors, will give a slight increase in backplane bus access clashes. This is due to the increased asynchronous nature of communication.

4.5.2. The need for distributed synchronisation in the parallel engine simulation.

The serial 'bottleneck' of the central synchronisation is a major impediment to increased speed and efficient processor utilisation. When increased parallelism is identified in the engine simulation the parallel structure of the simulation will become 'finer grained'. An approach based on central coordination of the engine simulation will increasingly limit the processing speed gains obtained by any extra parallelism. A distributed strategy for synchronisation of the diesel engine simulation is therefore important.

Distributed synchronisation is more complex than centralised synchronisation. All tasks in the simulation, rather than one central task, take part in the control of the simulations solution. Individual tasks in the simulation have to have synchronisation control structures for all other tasks with which they need to communicate. Communication requirements can however be better organised and controlled than for the centralised case. This leads to improved processor utilisation and therefore calculation efficiency. However it also slightly increases the amount of inter-processor communication due to synchronisation.

With distributed control of the engine simulation synchronisation can be made specific to need. Tasks can continue with calculation rather than waiting for everything to synchronise. This uses processing power more efficiently and increases calculation speed. It also improves the possibilities for efficient multi-tasking of several tasks onto one processor when processor numbers are limited, which multi-processor TRIPOS allows.

4.5.3. The need to avoid unnecessary inter-task communication in the engine simulation.

With the increased data communication, which will occur with the improvements outlined above, it is important that unnecessary data communication between simulation tasks is eliminated. For example, during half of the engine cycle a cylinder control volume has both of its valve connections closed. The cylinder task therefore will not need to communicate or synchronise with any manifold control volume tasks. To reduce communication and give efficient use of the computer backplane, tasks must have communication structures that allow the closing and opening of communication channels to other tasks. A structure where communication channels remain permanently open will mean the transfer of a large amount of redundant information.

4.5.4. The need to remove intermediate calculation from the engine simulation tasks.

In the redesign of the engine simulation, described in this chapter, a full analysis of the fundamental results from each engine calculation task has been carried out. Intermediate calculation, which has been used frequently in the previous parallel simulations, is only used where it saves repeated calculation of a variable. Where an intermediate variable aids program readability BCPL MANIFEST procedures are used. BCPL MANIFEST procedures are described in section 4.5.5.

The benefit of removing intermediate calculation is enhanced due to the redesign of the MC68020 BCPL compiler by Selwyn [45]. The new compiler uses the complete set of floating point registers on the MC68881 floating point co-

processor [28]. This allows intermediate floating point variables to be retained in the floating point registers rather than always being read in to the floating point co-processor for each calculation.

The new BCPL compiler also implements the MC68881 co-processor trigonometric and transcendental floating point instructions directly as BCPL functions rather than procedure calls. This allows improved register use as the results of these functions can remain in the co-processor registers rather than being transferred to the registers of the MC68020 processor or its local memory. The use of intermediate variables forces the storage of a variable into memory and so is best avoided.

4.5.5. The use of BCPL MANIFEST procedures in the engine simulation code.

BCPL is a block structured programing language. It is therefore good practice to use procedures to aid program readability. However procedure calls incur a time penalty. BCPL allows the use of MANIFEST procedures. With these the code of a procedure is inserted into the body of the program code rather than using a call to the procedure. This allows maximum speed whilst giving well structured and readable code.

4.5.6. The use of lookup tables in the engine simulation and their efficient use.

The use of lookup tables in the simulation improves calculation speed. Lookup tables have been used in the CSSC simulation and a list of these lookup tables is given in table 4.1. The lookup tables in the CSSC simulation are however used

inefficiently. Table offsets for the lookup tables are often calculated fully in floating point before being converted to fixed point. In the new parallel diesel engine simulation, developed for this work, improved lookup table routines have been used. In the new simulation conversion is made as soon as possible to fixed point in the calculation of the lookup table offset.

TABLE 4.1. A list of all the lookup tables used in the parallel diesel engine simulations.

Calculated variable.	**Reference variable(s).**	**Routines where used.**
Cylinder volume.	Cylinder crankshaft angle.	Pressure.
Cylinder surface area.	Cylinder crankshaft angle.	Heat transfer.
Cylinder rate of change of volume.	Cylinder crankshaft angle.	Output torque.
Cylinder inlet valve effective flow area.	Cylinder crankshaft angle.	Valve flow.
Cylinder exhaust valve effective flow area.	Cylinder crankshaft angle.	Valve flow.
$T^{0.4}$	Temperature. (T)	Heat transfer.
$V^{0.6}$	Volume (V)	Heat transfer.
$(l.\omega_e/\pi + 1.4)^{0.8}$	Engine speed (ω_e)	Heat transfer.
Critical pressure.	Ratio of specific heats	Valve flow.
$T^{0.5}$	Temperature (T)	Valve flow.
$R^{0.5}$	Gas constant (R)	Valve flow.
Choked flow parameter	Ratio of specific heats	Valve flow
Non choked flow parameter	Ratio of specific heats and pressure ratio.	Valve flow

Although the use of lookup tables improves simulation speed they also effect the accuracy of the simulation. The use of lookup tables was particularly important when the diesel engine simulation used the MC68000 processor. The MC68000 has no hardware floating point capability. However, with the MC68020/MC68881 processor board floating point calculation is far more efficient. For this reason, the new simulation uses both lookup tables and direct computation for the calculation of cylinder valve flows and cylinder wall heat loss. The user is able to select the style of calculation wanted. The calculation of cylinder volume, surface area and the rates of change of volume however remain calculated by lookup tables. No inaccuracy is incurred through the use of these lookup tables. This is because the integration always occurs on 0.25^{o} boundaries, which is the accuracy of the lookup tables.

4.6. A new diesel engine simulation.

For the research described in this thesis, the parallel computer multi-volume diesel engine simulation has been completely redesigned. The new design incorporates all the design criteria outlined in section 4.5. The new simulation has distributed control and communication between tasks at each iteration step. It is described in the thesis as a distributed iteration step communication (DISC) simulation. The distributed nature of the simulation is explained in the following sections.

4.6.1. An overview of the new DISC parallel diesel engine simulation.

The 'filling and emptying' method views a diesel engine as a series of interconnected control volumes. To solve the differential equations of the control volumes, the modified Euler predictor corrector formulae are used. This is used

iteratively until a stability criterion is met. Communication between the volumes has to occur for each iteration of the formulae. This iteration must continue, until all the control volumes that are connected together are stable. The solution may then progress. In the CISC implementation of the parallel diesel engine simulation this iteration is controlled by a central synchronisation task. In the new DISC simulation, the control of the simulation is a part of all the simulation tasks.

The new simulation has no central synchronising task. Instead all the simulation tasks communicate directly with one another. The progression of the simulation is controlled by the combination of all the inter-task synchronisation. The tasks are synchronised with other tasks, when data needs to be transferred from one to the other. The new parallel TL11 diesel engine simulation consists of six cylinder volume tasks, three manifold control volume tasks, a dynamics task and an actuator task. To these is added a system stability task which is described in section 4.6.3.

4.6.2. An overview of synchronisation and communication in the DISC parallel diesel engine simulation.

The inter-task synchronisation in the new simulation uses the TRIPOS communication primitives QPKT and TASKWAIT. All communication between tasks in the simulation is two way, with both tasks using data from each other in their calculations. A list of these inter-task data transfers is given in table 4.2.

TABLE 4.2. A complete list of all the communication between tasks in the DISC diesel engine simulation.

Communicating tasks.	Type of communication.
Cylinder to manifold.	Valve flow parameters. Mass, fuel air ratio, and enthalpy.
Cylinder to dynamics.	Cylinder torque.
Cylinder to actuator.	Simply returns communicating packet.
Cylinder to stability.	Cylinder control volume stability flag TRUE,FALSE.
Manifold to cylinder.	Manifold state to enable the cylinder control volume to calculate valve flow.
Manifold to dynamics.	Turbocharger or compressor torque.
Manifold to actuator.	The exhaust manifold simply returns communicating packet.
Manifold to stability.	Manifold control volume stability flag TRUE,FALSE.
Dynamics to cylinder.	Engine speed.
Dynamics to manifold.	Turbocharger speed and engine speed.
Dynamics to actuator.	Engine speed.
Actuator to cylinder.	Cylinder control parameters. Mass of injected fuel and static injection timing.
Actuator to manifold.	Variable geometry turbocharger turndown value.
Actuator to dynamics.	Crankshaft load.
Stability to cylinder.	Simulation system stability flag. TRUE or FALSE.
Stability to manifold.	Simulation system stability flag. TRUE or FALSE.

For each of the two way inter-task data transfers, an inter-task communication packet is allocated. This TRIPOS packet is sent between the tasks and controls the reading and writing of data between the tasks. The sending of the packet by a task, indicates that the data results on the sending task may be read by the receiving task. The receiving of the packet indicates that a task may read the result area of the other task. The receiving of a packet also indicates that the other task has read its result area and that this may now be overwritten. Because of the control of the reading and writing by a packet there is no need for the double buffering of the engine task result areas which was described earlier.

The communication in the new system is designed to allow maximum processor utilisation. This is achieved by making maximum use of overlapped communication and calculation. To ensure this, the communication packet is sent from one task to the other task, immediately after all the data on the sending task is calculated. The waiting by a task for the communication packet from another task is not carried out until just before the data from the other task is needed. To allow this asynchronous communication, all of the engine simulation tasks have TRIPOS packet handling routines. For each communication packet, a packet flag is maintained on both communicating tasks. The packet handling routines set the communication flag on receipt of a packet. The flag is cleared when the communication packet is sent. The waiting task is only forced to suspend and wait for the packet if the communication flag is not set. The packet handling routines allow the asynchronous receiving of packets.

Three types of inter-task communication are used in the new simulation:

- communication between tasks for each predictor corrector formulae iteration,

- communication between tasks for each calculation step,

- communication between tasks at specific sections of the engine cycle.

Each type of inter-task communication is outlined in the following sections which describe the actions of each DISC simulation task.

4.6.3. The system stability task in the DISC simulation.

For the new DISC parallel diesel engine simulation a new type of task has been designed. The new task calculates system stability from the stability results from all the engine control volume tasks. The computation of system stability is purely a calculation function rather than being combined with the simulation control which was the case with the CSSC and CISC simulations. On each iteration of the corrector formulae on the engine control volumes the stability task is sent the result of each control volume's stability calculation. The stability task determines if all the control volumes in the simulation are stable and returns the system stability to the control volume tasks. The control of the iteration and possible steplength adjustment is distributed across all the control volumes. Each control volume task determines the progression of its solution from the system stability. The stability task has no direct control of the simulation solution.

4.6.4. The cylinder control volume tasks in the DISC simulation.

The new simulation structure incorporates all the data control structures for the cylinder control volume into the cylinder volume task. The cylinder control volume task has to communicate with the following tasks:

- the inlet manifold task,
- an exhaust manifold task,
- the shaft dynamics task,
- the engine actuator task,
- the system stability task.

The communication between the cylinder and manifold control volume tasks occurs for every iteration of the predictor corrector formulae. This communication is dealt with in more detail in section 4.6.9. The communication between the cylinder control volume and the engine dynamics task occurs at every calculation step. Communication with the actuator task occurs only during the compression period prior to the injection of fuel. This is because the cylinder control volume task only needs these values just prior to fuel injection.

4.6.5. The manifold control volume tasks in the DISC simulation.

The manifold control volume has to communicate with the following tasks:

- all of the connected cylinder control volume tasks,
- the dynamics task,
- the system stability task.
- engine actuator task (exhaust manifold only) to obtain the value of turbocharger variable geometry turndown.

The manifold task communicates with the engine dynamics task to obtain the turbocharger rotational speed. It also reads the engine crankshaft speed which it uses to determine the integration time step from the crankshaft angle step. The communication between the cylinder and manifold control volumes is described in more detail in section 4.6.9.

4.6.6. The engine dynamics task in the DISC simulation.

The engine shaft dynamics task calculates the speeds of the engine crankshaft and the turbocharger rotor from the torques generated by all the engine control volumes. It therefore has to communicate with all the cylinder and manifold control volume tasks. Because of the high integration time constant of the dynamics equations communication only needs to occur once for each calculation step. The dynamics task also communicates with the actuator task as the actuator task uses the engine speed to calculate its integration time step.

The equations for the shaft dynamics have a high integration time constant. They are therefore solved with a simple explicit integration algorithm, equation 4.3.

$$y_{n+1} = y_n + hf_n \qquad \text{Equation 4.3}$$

There is therefore no need to synchronise the dynamics task with the system stability task. The dynamics task is kept in step with the rest of the engine simulation through its communication with the control volume tasks at every calculation step. The engine simulation control volumes send their piston or turbine torque results at the end of each calculation step. The dynamics task returns the engine and turbine speed to the control volume tasks at the

beginning of each calculation step.

4.6.7. The engine actuator task in the DISC simulation.

The engine actuator equations, like the dynamics equations, have a high integration time constant. They are therefore solved for each integration step independently of the integration iteration of the control volume tasks . The control volume tasks send the actuator task a communication packet. This packet is updated with the newly calculated engine control parameters and is then returned. The communication with the exhaust manifold task of turbine variable geometry occurs at every calculation step. The communication with the cylinder control volume task of injected fuel mass and injection timing only needs to occur prior to the injection of the fuel. Therefore each cylinder control volume task communicates with the actuator task at each calculation step during the compression sector.

The actuator task is kept synchronised with the rest of the simulation through its communication with the engine dynamics task. The actuator task reads the engine speed from the dynamics task. This allows it to calculate the integration time step from the integration crankshaft angle step.

4.6.8. Communication between the control volume tasks and the stability task.

The system stability calculation allows concurrent communication and calculation with the control volume tasks. All the control volume tasks may continue calculating the control volume state gradients after calculating the volume stability. This calculation can be carried out before the decision has to be made,

based on the system stability, of whether to reiterate or to proceed to the next calculation step. Figure 4.5 illustrates this overlapping of calculation and communication.

To reduce the need for communication, the stability task has special structures to handle closed cycle cylinder control volume tasks. During the closed half of the engine cycle the cylinder control volumes are not connected to any other control volumes. They therefore do not need to synchronise the iteration of their predictor corrector routines. During the closed half of their cycle the cylinder volume tasks can calculate the volume state without reference to the system stability task. The purpose of the stability task is to control the iteration of connected control volumes.

To allow simple control of the stability task, when the disconnected volumes reconnect, the closed cylinders are made to communicate with the stability task at the beginning of each calculation step. All control volumes receive a stability packet at the beginning of a new calculation step. The closed cycle cylinder volume tasks return this immediately indicating that they are unconnected. The stability task then does not resend any stability packets associated with a closed cycle cylinder volume until the next integration step. The stability packet is sent to all the other control volume tasks on every iteration of the predictor corrector. This iteration continues until system stability is reached.

4.6.9. Communication between the cylinder control volume tasks and the manifold control volume tasks.

During the engine cycle, valves between cylinders and manifolds open and close. In the time that a cylinder valve is closed the manifold and the cylinder control volume tasks do not need to exchange data or to synchronise with each other. Making the two volumes synchronise in this time is wasteful of computer backplane communication bandwidth. The ability to open and close communication channels, between the cylinder and manifold control volume tasks, is therefore important. To allow this, an enhanced synchronisation structure has been designed for the cylinder to manifold communication. This communication structure is outlined in the following sections.

4.6.9.1. Detection of cylinder calculation routine change.

The point at which cylinder valves open and close is determined by the engine crankshaft angle. The crankshaft angle therefore determines when the cylinder-manifold communication channels must close and open. The manifold control volume tasks have no absolute knowledge of the crankshaft angle. Therefore, the detection of the opening and the closing of the cylinder valves is done by the cylinder control volume tasks. The opening and closing of valves also affects the cylinder control volume equations. Therefore the detection of the valves opening and closing is done in conjunction with the detection of the cylinder equation changes.

The differential equations of the simulation, on all the control volume tasks, are solved using an integration formula with a fixed initial steplength. The position of the start and finish of the integration steps, and the angle at which the cylinder control volume equations change do not necessarily coincide. However, in the transition between cylinder sectors the control volume equations become either a simpler subset of the previous cylinder equations, or have added terms compared to the previous equations. Therefore the more complicated set of equations are used for the integration step containing the sector transition. For example in the transition from scavenge to induction and the transition from exhaust to scavenge the more complex equations for scavenge are used. Valves are therefore presumed to close at the end of the integration step containing the valve closure angle. Valves are presumed to open at the beginning of the integration step containing the valve opening angle.

4.6.9.2. Communication control structures on the manifold and cylinder control volume tasks.

A manifold control volume task may be connected to any number of cylinder control volume tasks. It therefore has to handle a variable number of cylinder communication channels. The manifold control volumes have no local knowledge of when cylinder valves will open or close. The cylinder control volume tasks therefore must inform the manifold control volume tasks of the opening and the closing of the interconnecting valve. This also implies the opening and closing of the common communication channel.

The cylinder control volume controls the communication between the cylinder and the manifold control volumes using three types of TRIPOS packets:

- A communication packet,
- A cylinder valve close packet,
- A cylinder valve open packet.

The operations of these are detailed in sections 4.6.9.3 to 4.6.9.5. For the control of the cylinder communication channels the manifold task maintains the following structures:

- A table for the storage of the cylinder communication packets,
- A count for the number of connected cylinders,
- A count of the number of cylinder communication channels that will open on the next integration step,
- A count of the number of communication channels that will close on the next integration step.

The purpose of each of these is explained in the sections 4.6.9.3 to 4.6.9.5.

4.6.9.3. The normal communication between a cylinder and a manifold control volume task.

For each cylinder to manifold connection in the simulation there is a communication packet. This is used for the coordination of the reading of data by one task from the other task. The cylinder control volume task reads the state of the manifold control volume at the beginning of each integration iteration. The cylinder control volume uses the manifold state to calculate the

interconnecting valve flow parameters of mass flow, fuel-air ratio and enthalpy. These values are then read by the manifold control volume task.

It should be noted that the cylinder manifold communication scheme gives good concurrency. Whilst the cylinder volume task calculates valve flow, the manifold volume task can calculate the flow through the turbine or compressor.

Figure 4.6 illustrates the sequence of actions that occur for each iteration of the integration formula on the manifold and cylinder control volumes.

4.6.9.4. The communication protocol for closing the communication channel between a cylinder and a manifold task.

The closing of the communication channel between a cylinder and manifold control volume is relatively straight forward. The cylinder volume task sends the manifold task a cylinder close packet. This packet is sent during the integration step prior to the closure of the interconnecting valve. The manifold task counts the number of cylinder valve close packets that arrive. From this it determines the number of valves that will close on the next integration step. This is used to update the count of connected cylinders on the manifold control volume at the beginning of the next integration step. No special care has to be taken with the communication of the cylinder close packet. This is because the two control volumes will synchronise using the standard communication packet after the sending of the cylinder valve close packet. This ensures that at the beginning of the next integration step both control volumes are aware of the communication channel closure. A block description of the actions by both cylinder and manifold on closing the communication channel between them is given in figure 4.7.

4.6.9.5. The communication protocol for the opening of the communication channel between a cylinder and a manifold task.

With the opening of a communication channel between a cylinder control volume and a manifold control volume there is no direct synchronisation between the two tasks. Therefore, unlike in the case of the cylinder valve close procedure, there is no direct synchronisation between the tasks which can ensure the correct interpretation of the cylinder valve open packet by the manifold volume task.

The opening of the communication channel between the cylinder and the manifold control volume tasks has to be made with reference to a task with which both tasks synchronise and communicate. Both tasks communicate with the system stability task. The cylinder control volume task by sending the valve open packet to the manifold task before communicating with the stability task ensures that the opening of the communication channel is determinant.

If the communication channel is opened without reference to the stability task communication problems will occur. It is possible for the manifold control volume task to receive the valve open packet after the manifold task has received all the communication packets for the next step. The manifold control volume would then proceed to predict its new state with no reference to the newly opened cylinder valve. The manifold control volume would lose communication packets and these would not be returned to the cylinder task. This would cause the cylinder tasks and thus the simulation to stop. A block description of the actions by both cylinder and manifold on the opening of the communication channel between them is given in figure 4.8.

4.7. An analysis of the execution speed of the new DISC simulation.

Parallel simulation of a diesel engine offers increased speed of processing. This section gives an evaluation of the speed of processing offered by the new DISC simulation method. Comparison is made between the results given by Jones [8] for a CSSC simulation, an improved CSSC simulation developed in the initial research for this thesis and the DISC simulation.

4.7.1. An analysis of improved simulation coding through a comparison of the original and the developed CSSC simulations.

The results given by Jones for a TL11 engine simulation were for the CSSC type of simulation. The CSSC type of simulation was also used in the initial research for this thesis. In this initial research, the CSSC simulation was re-coded to improve its efficiency. This involved the removal of intermediate variable calculation, efficient lookup table use, the removal of inter-task communication through the synchronisation task, and the implementation of the double buffered direct communication as outlined in section 7.3. The improvement in execution time due to more efficient coding of the simulation routines and the use of the more efficient compiler can be estimated by comparing the simulation speeds for a single cylinder engine model on one MC68020 processor before and after the coding improvements. Before the improvements a simulation speed of 80rpm was possible. After the improvements a simulation speed of 181.87rpm was obtained, a speed up of 2.25.

The improved CSSC simulation was implemented using twelve MC68020 processors. This allowed a processor allocation of one processor per simulation task. Using this a simulation speed of 60 rpm for a simulated engine speed of 1500 rpm was obtained.

Jones in his thesis [8] estimated a simulation speed of 10 rpm for the TL11 simulation using one MC68020 processor per control volume. This estimation assumed the use of 25MHz processors and made no allowance for communication overhead due to bus access clashes. With the 16.667MHz MC68020 processors used for this research a better estimate would be a simulation speed of 6 rpm for the simulation developed by Jones.

A significant improvement in processing speed for the CSSC simulation was shown in this initial research. It is estimated that 50% of this increased simulation speed was due to increased processing speed using the new BCPL compiler. The rest of the increase in speed was due to increased efficiency in calculation and communication.

4.7.2. An analysis of computational performance for the DISC diesel engine simulation.

The use of the CSSC simulation is computationally inaccurate. The initial development of the CSSC implementation of the engine simulation was therefore further developed into the fully distributed DISC style of simulation described in section 4.6. If a method based on the CISC simulation had been developed, a slower simulation speed than that for the CSSC simulation would have been achieved. This would be due to the increased usage of the central synchronisation task both for synchronisation and communication. The CISC

simulation would also be slower as the engine simulation tasks would synchronise and communicate more than once for each calculation step.

The use of the distributed control DISC simulation gives the same simulation accuracy as the CISC type of simulation. Therefore, like the CISC style of simulation the amount of inter-task communication for each calculation step is greater than that for the CSSC type of simulation. However using the DISC type of simulation the simulation speed has been kept at about 60 rpm using eleven MC68020 processors. This has been achieved through greater computational efficiency. When the task utilisation for the new method DISC simulation is monitored the reason for the increased simulation speed is readily apparent. All the simulation tasks use processing power very efficiently. Most of the processors in the parallel computer are being used at around 80% utilisation. This compares with the average value of processor utilisation of 40% which is obtained when the centrally synchronised CSSC simulation is used. The maintaining of simulation speed with increased simulation complexity is due to the improved communication structures in the DISC simulation. These allow better use of all the processor's computing power and this is demonstrated in the approximate doubling of processor utilisation for the DISC simulation compared to the CSSC simulation.

It should be noted that the increased efficiency of the DISC simulation causes a problem. With the increased simulation throughput the amount of backplane usage is greatly increased. Chapter 3 describes the hardware problems associated with the use of the MC68020 in the BUPC backplane. The high communication rate which the new simulation needs causes processor error. However with the maximum of eleven MC68020 processors it is possible to obtain a reasonable length of simulation and the results for simulation speed can be obtained.

Using the new DISC simulation, an analysis of the effect of various calculation parameters on simulation speed has been carried out. The parameters that have been varied are given in table 4.3. The results from the analysis are given in graphical form in figures 4.9.

TABLE 4.3. The parameters varied as part of the analysis of the parallel diesel engine simulation.

Engine speed	1000 rpm - 2100 rpm
Integration steplength	0.5^{o} - 2.0^{o}
Integration stability criterion	0.1% and 0.5%

In figure 4.9a the results for the engine simulation speed against initial integration steplength are given. These results were obtained using eleven MC68020 processors with an allocation of one processor per engine control volume. The integration was achieved using a stability criterion of 0.5%. The results show a simulation speed maximum of 56rpm with an initial steplength of 2^{o}. Above 2^{o} the time saving due to increased integration steplength is lost due to the increased need for steplength reduction in areas of integration instability.

Figure 4.9b shows the effect of the simulated engine speed on the speed of the engine simulation. These results were obtained using eight MC68020 processors with two cylinder control volumes on each processor and one processor each for the manifold control volumes. Three sets of results are shown in figure 4.8b for the initial integration steplengths of 0.5^{o}, 1.0^{o} and 2.0^{o} all with a stability criterion of 0.1%. The results show that between 1000 rpm and 1500 rpm the optimum integration steplength is 1o and above 1500 rpm it is 2o. The critical

parameter in the stability of integration, for the engine simulation, is the integration time steplength rather than the size of the crankshaft angle integration step.

Figure 4.9c shows the effect on simulation speed of varying the integration steplength with two different integration stability criteria of 0.1% and 0.5%. The results indicate that for an initial steplength of 0.5^{o} that there is little advantage in the wider stability margin. With the increase in initial steplength to 1^{o} the simulation speed for the 0.5% criterion approximately doubles due to a halving in the number of integration steps. However with the 0.1% stability criterion this is not achieved due to an increase in integration steplength reduction in areas of instability. For the 0.5% stability criterion increased speed is obtainable between 1^{o} and 2^{o} however here there is increased integration steplength reduction.

4.8. Conclusion.

A new parallel diesel engine simulation has been developed. This new simulation has a distribute structure and therefore has no centralised controlling task. The new structure gives better utilisation of all the processors in the parallel computer and a higher simulation speed than the previous parallel simulation. The new parallel simulation is also an exact representation of the 'filling and emptying' method. Using the new parallel simulation a simulation speed of 56 rpm has been achieved or 26.78 times 'real time'.

Figure 4.1. The TL11 engine parallel computer simulation structure illustrating the communication between the simulation tasks.

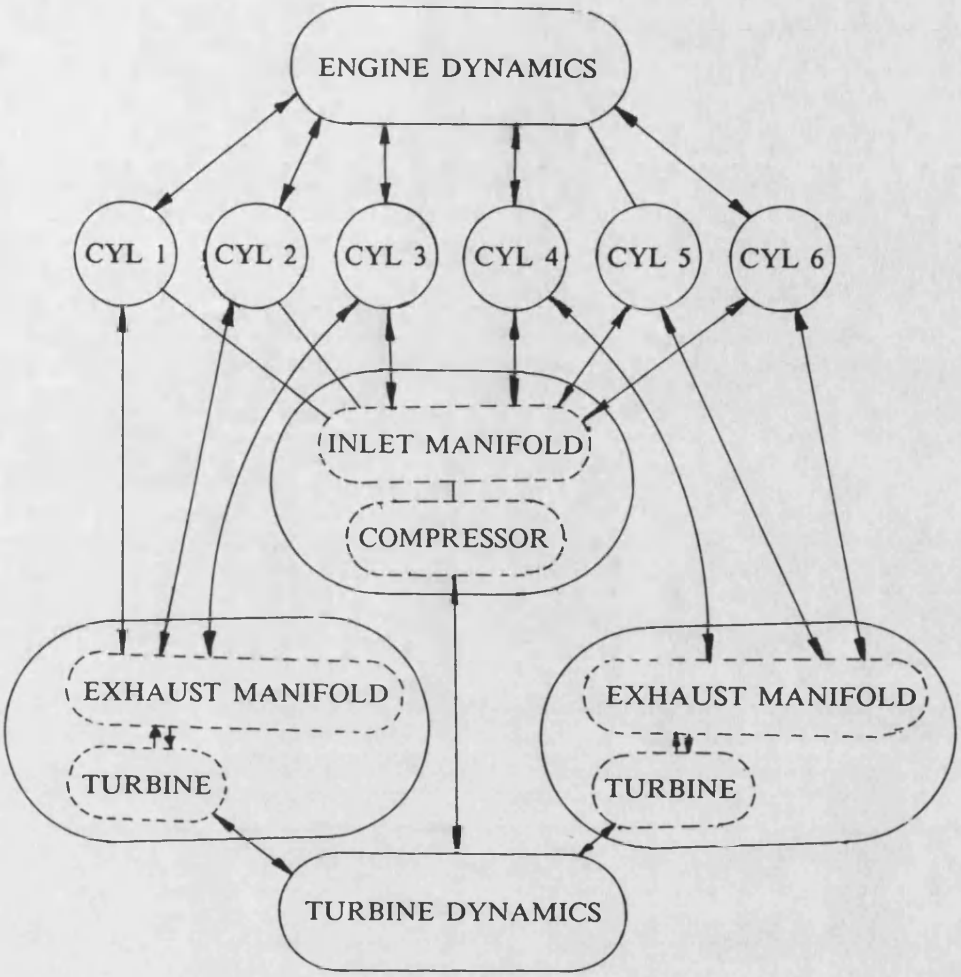


Figure 4.2. A diagram illustrating the application of the modified Euler predictor corrector method.

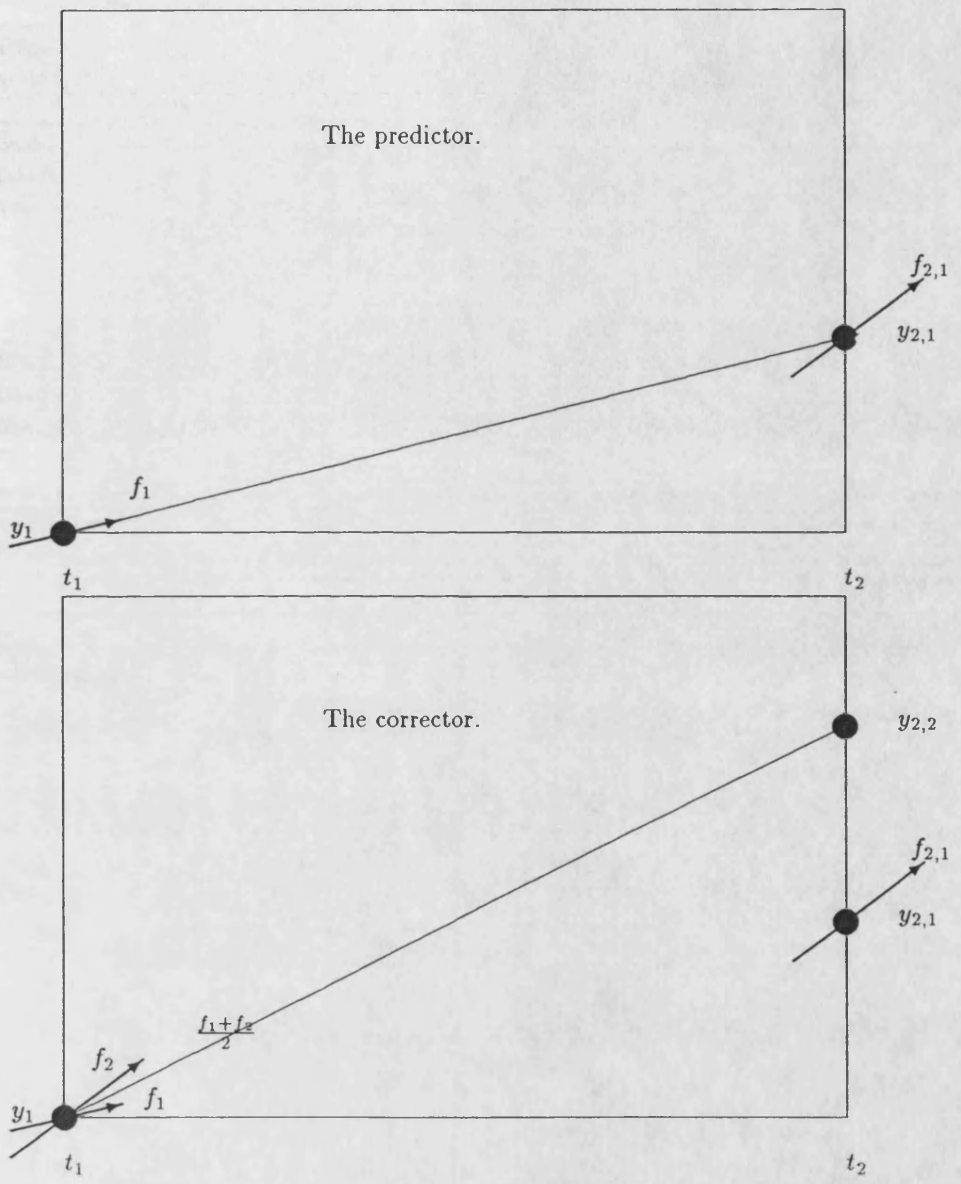


Figure 4.3. A communication and calculation flow diagram for the CISC simulation.

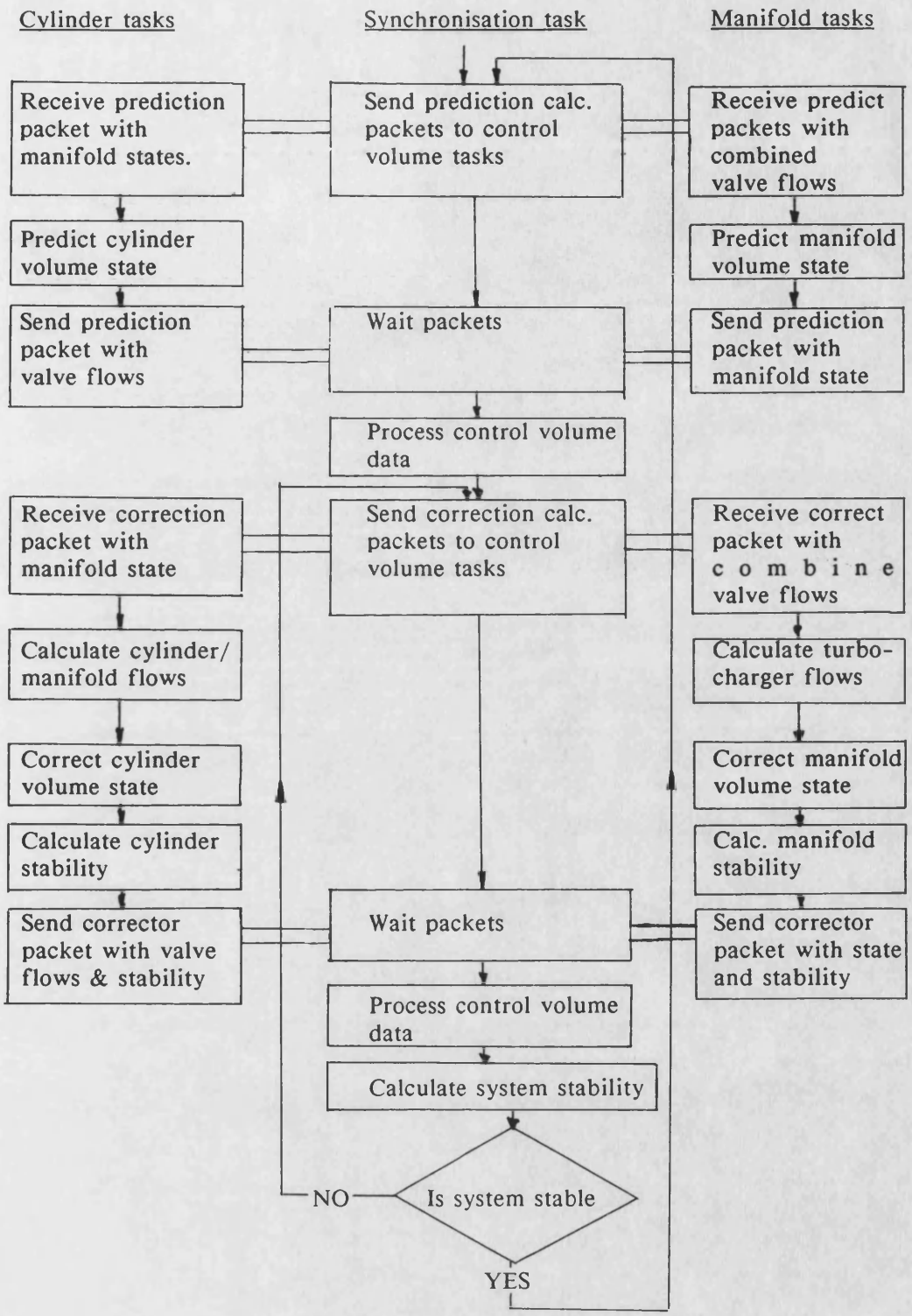
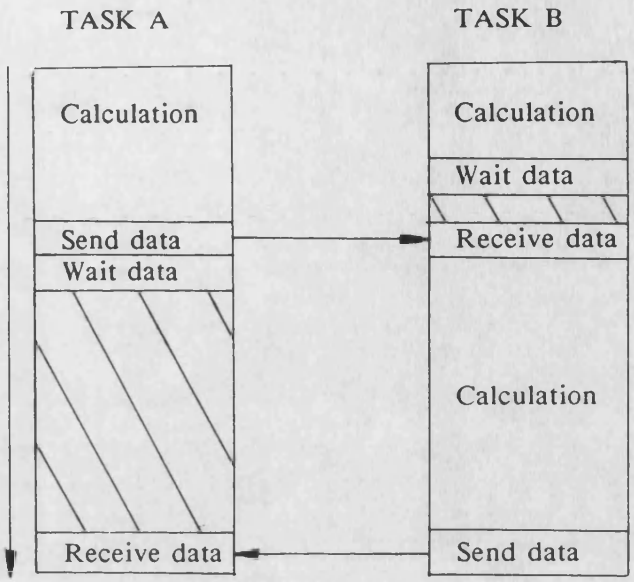


Figure 4.4. An illustration of overlapped and non-overlapped communication and calculation.

Non overlapped communication and calculation.



Overlapped communication and calculation.

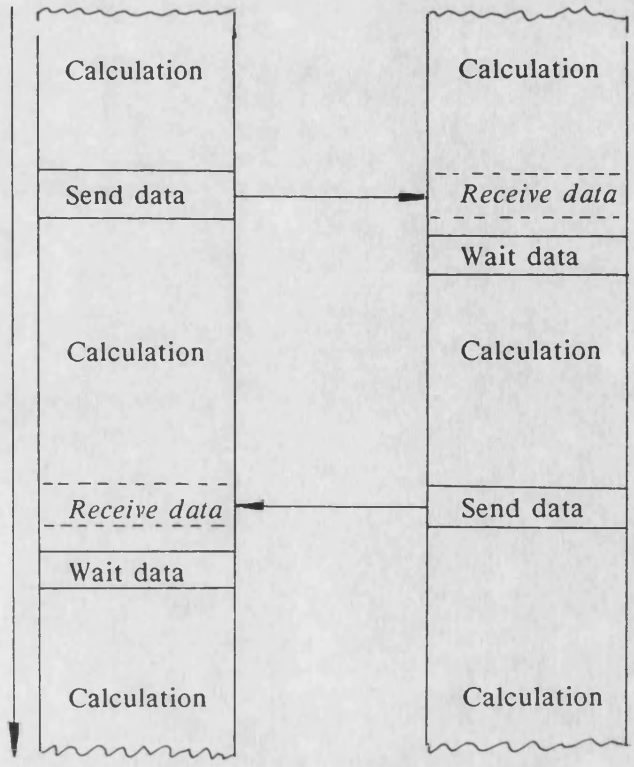


Figure 4.5 The overlapping communication and calculation between the engine control volume tasks and the system stability task.

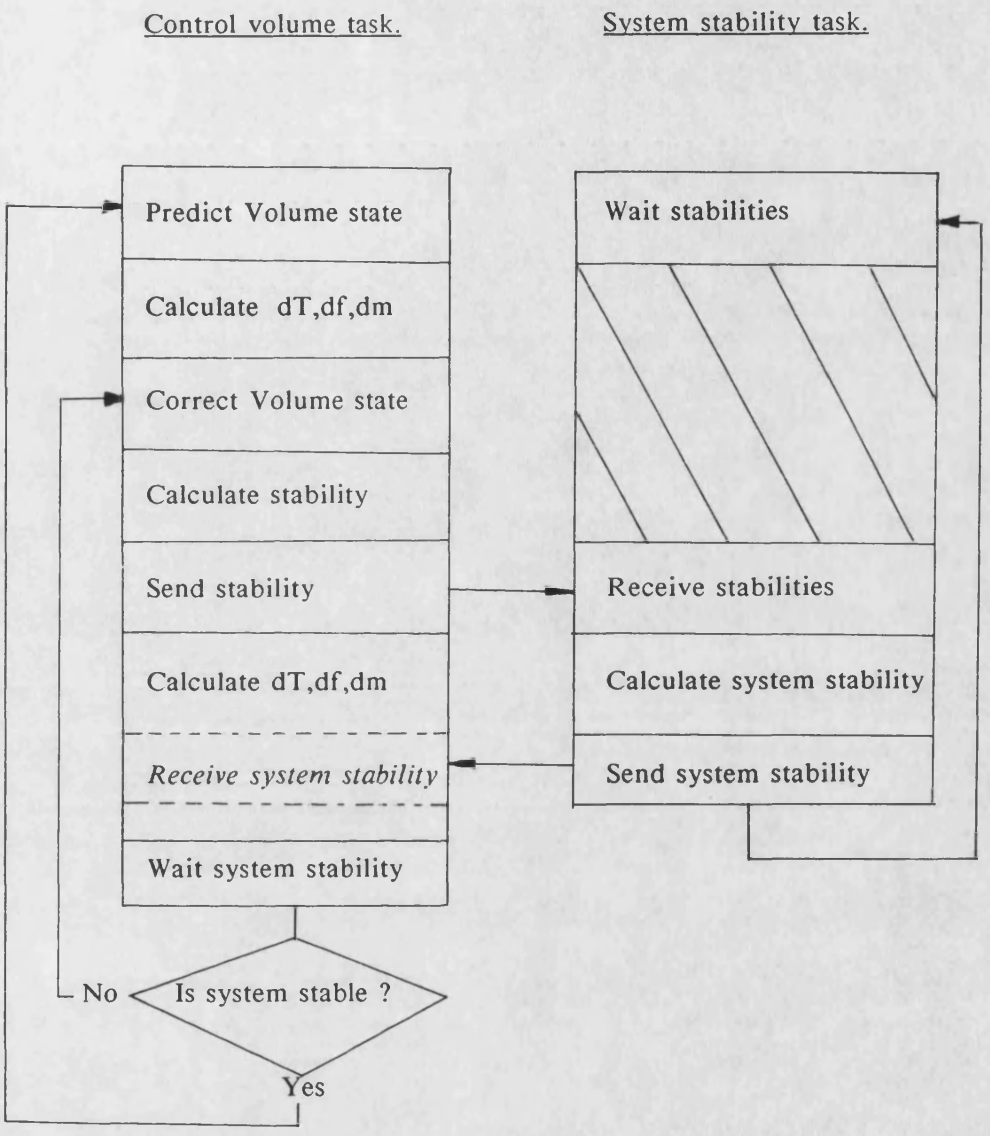


Figure 4.6. A block diagram of the actions involved in using the communication channel between a cylinder control volume task and a manifold control volume task.

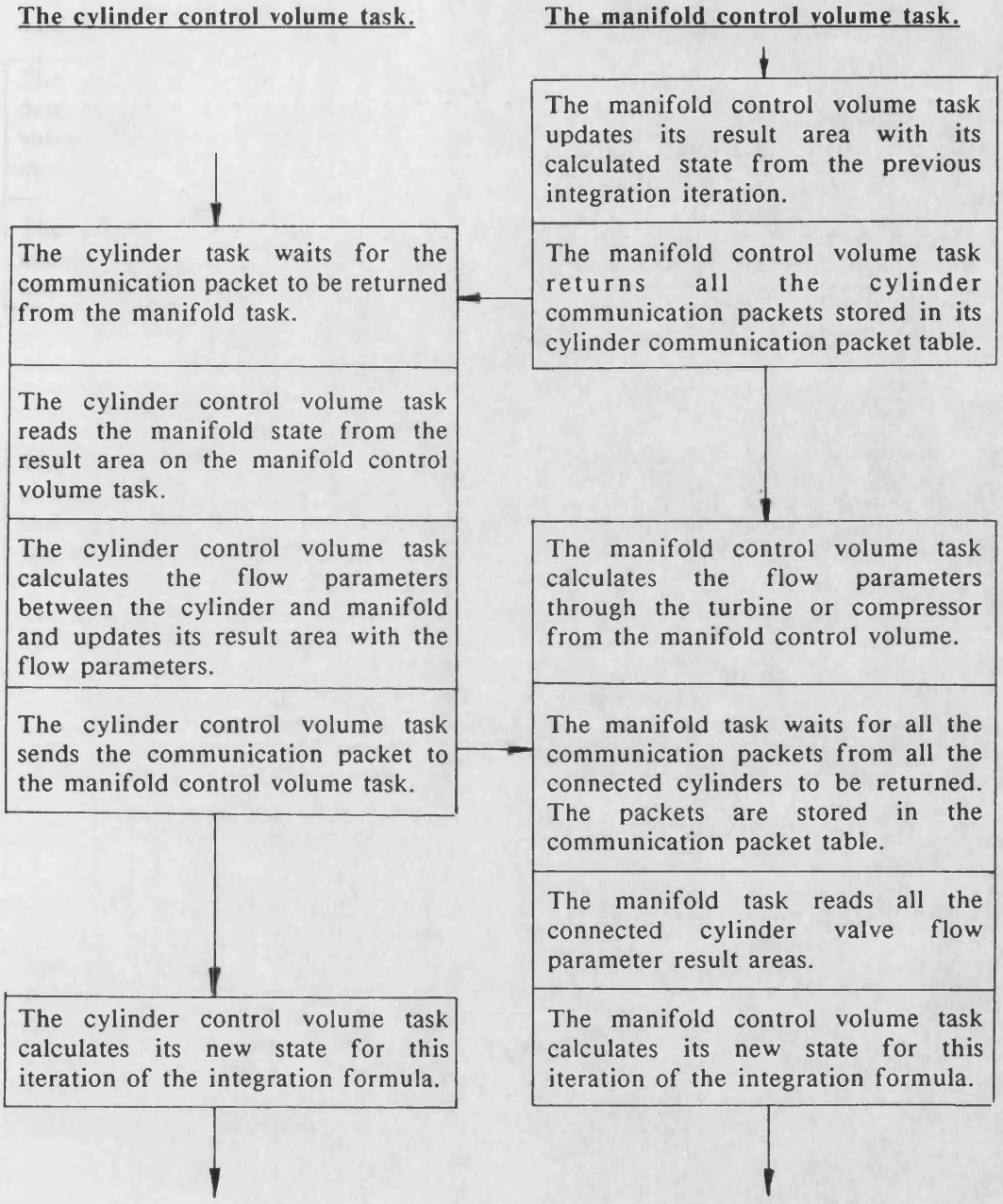


Figure 4.7. A block diagram of the actions involved in closing a communication channel between a cylinder control volume task and a manifold control volume task.

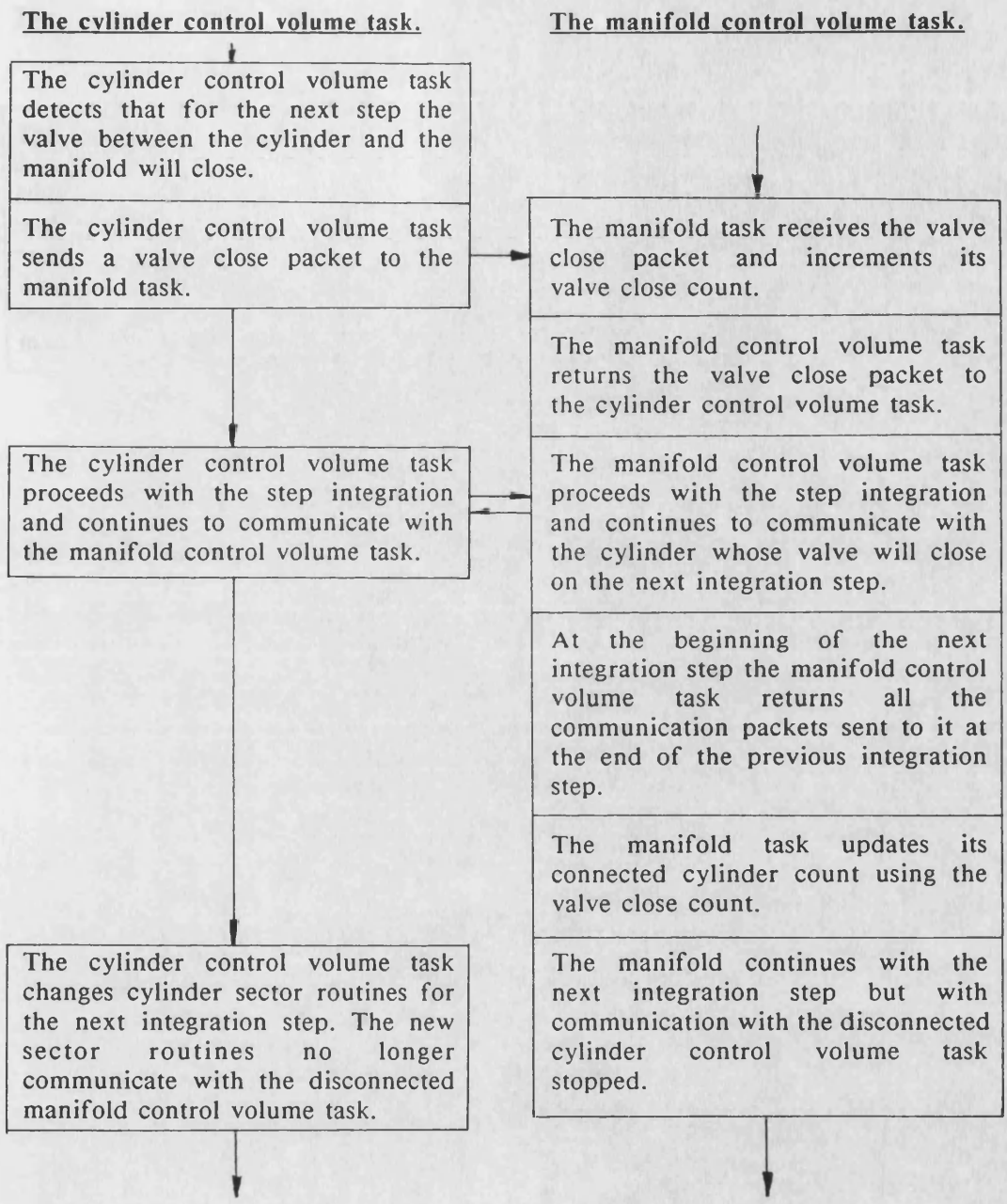


Figure 4.8. A block diagram of the actions involved in opening a communication channel between the cylinder control volume task and a manifold control volume task.

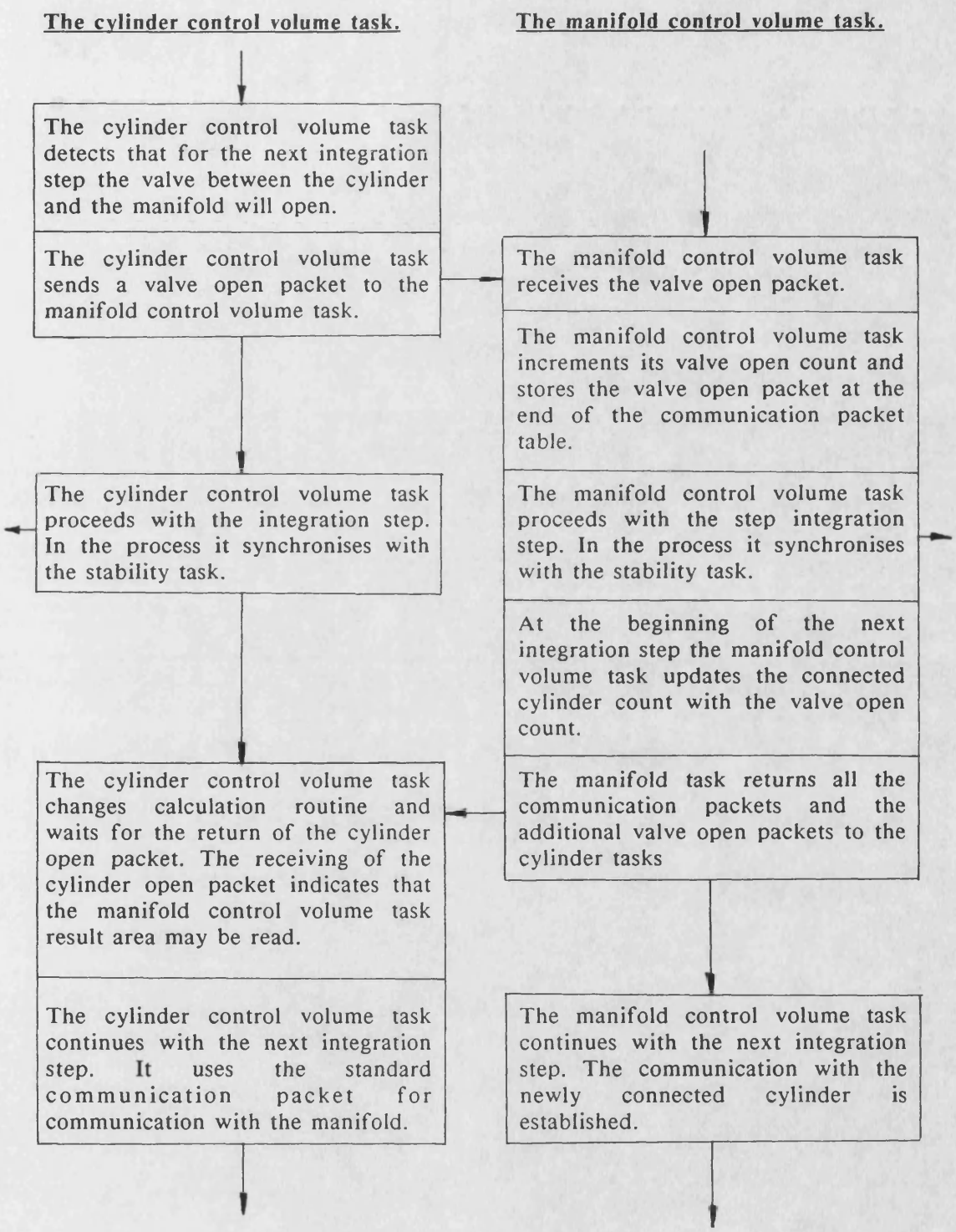


Figure 4.9a The effect of initial integration steplength on simulation speed for a stability criterion of 0.5%.

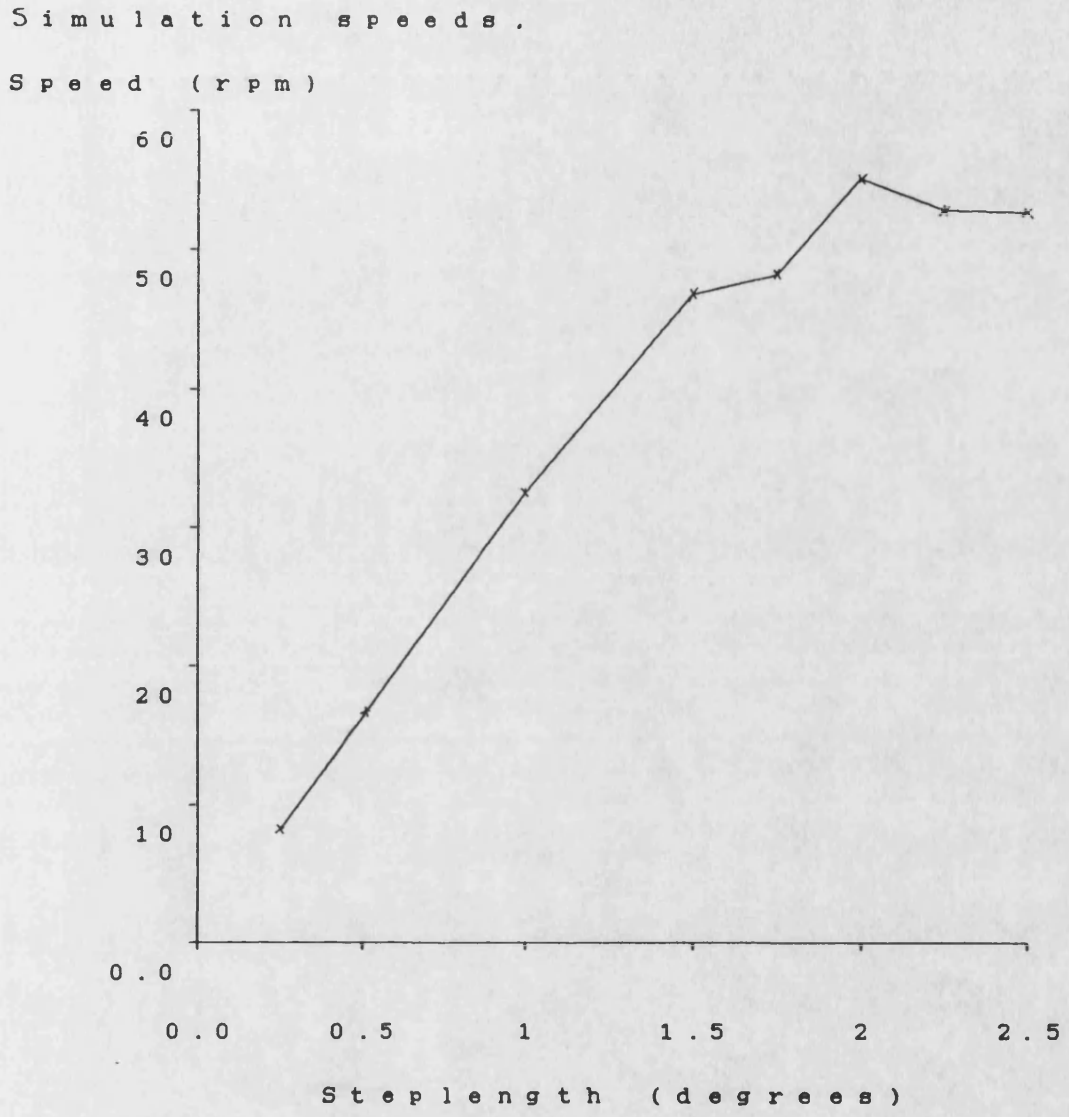


Figure 4.9b The effect of simulated engine speed on simulation speed for different initial integration steplengths.

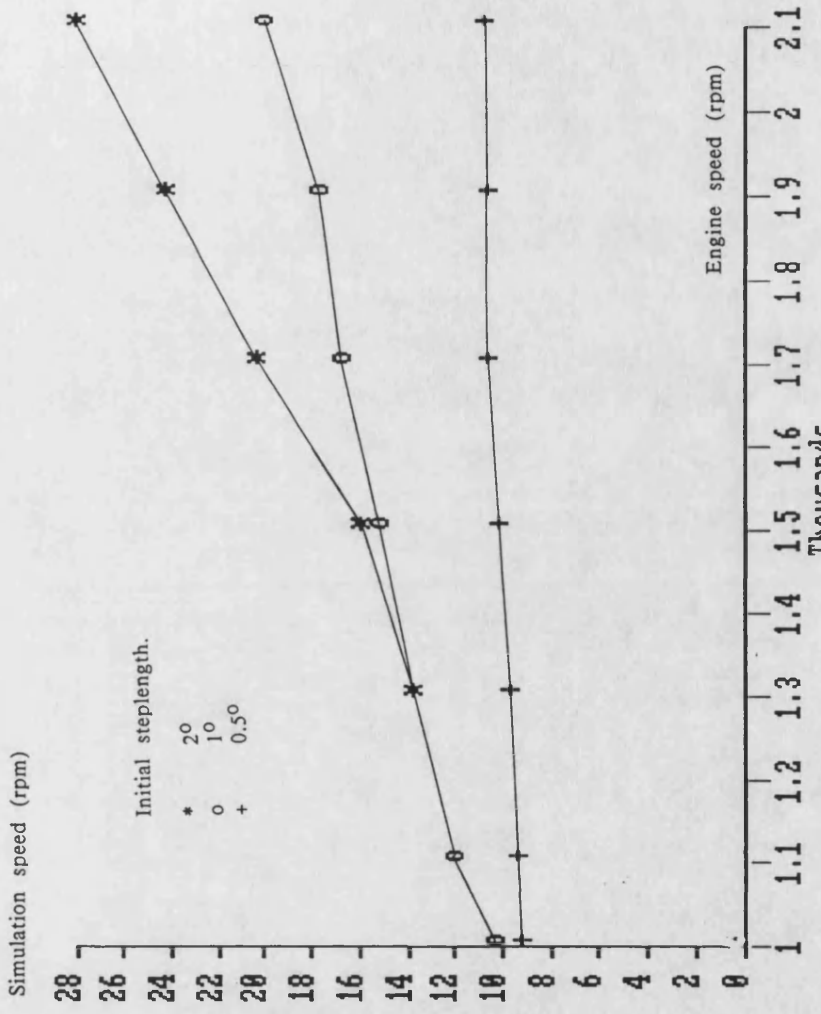
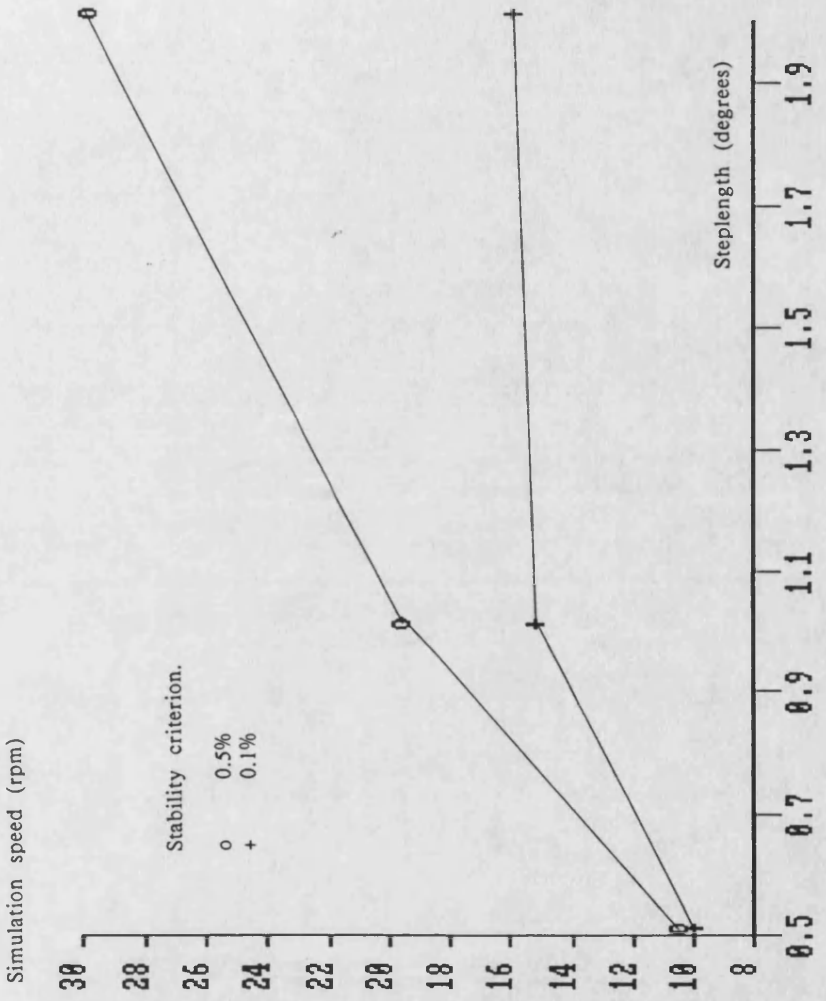


Figure 4.9c The effect of integration steplength and integration stability criterion on simulation speed, for a 1500 rpm engine speed.



CHAPTER 5.

THE USER INTERFACE TO THE DIESEL ENGINE SIMULATION.

5.1. Introduction.

As described in chapter 4 the new parallel diesel engine simulation was developed from the simulation produced by Jones [8]. Initially the user interface and the graphical display from the Jones simulation were used for the new parallel diesel engine simulation. This chapter describes the changes that were made to the user interface and the graphical display. The last part of the chapter describes the methods used for the recording and displaying of the results from the simulation.

5.2. The development of a menu based interface for the parallel diesel engine simulation.

The user interface developed by Jones was based on a command line interpreter which used the BCPL RDARGS library routine [34]. The user therefore selected the simulation interface routines by typing in command lines. Two reasons prompted the development of a new user interface for the simulation. These were that the user interface could be made more user friendly and that the computer terminal could be used for other things apart from command selection. For example the computer terminal could be used for the continuous display of the results from the simulation.

The use of the line interpreter based on RDARGS precludes the use of the computer terminal for the continuous display of results. Continual display of results is therefore only possible using the graphical display of the simulation. The graphical display of the simulation is described in the next section. By developing a terminal interface which used the TRIPOS single character mode [35] rather than the line interpreter the whole of the terminal screen can be used for the display of information. This allows the display of simulation results

and the development of a menu based user interface. In the TRIPOS single character mode, each character written and read from the terminal by TRIPOS is treated separately. The terminal input and output is usually dealt with on a line by line basis by TRIPOS.

A menu based user interface has therefore been developed for the parallel diesel engine simulation. This has been based on the TRIPOS screen editor ED which gives the basic screen reading and writing functions as well as the screen cursor movement commands. It also provides the basic task communication structures for using the computer terminal in single character mode. Using ED as a basis, a set of basic menu functions were designed based on a standard terminal screen layout.

The terminal screen layout for the new user interface for the parallel engine simulation consists of two distinct display areas. It consists of one large display area at the top of the screen for the display of results or menu options and a smaller one line display area for the display of menu options or menu comments. The use of the two display areas allows one area to be used for the display of menu options to the user whilst the other area displays information or simulation results.

The terminal used in the research for this thesis has been a Televideo TVI905. This has some specialist screen commands which the standard TRIPOS terminal interface does not handle. These commands allow the protection of areas of the the terminal screen from subsequent overwriting. These have been used to improve the appearance of the menu output by framing the two menu information areas. To maintain the general purpose nature of the menu for other terminals these TVI905 terminal specific commands have been made conditionally compiled.

A full list of the basic menu development functions and their purpose is given in appendix D. The design of a standard set of menu development functions allows the menu interface to be easily expanded and adapted. Figures 5.1a and 5.1b show some of the menu screens produced for the diesel engine simulation. Figure 5.2 gives the full structure of the diesel engine simulation user interface menus.

5.3. An animated graphics display for the diesel engine simulation.

5.3.1. The original animated display.

The original graphical display for the diesel engine simulation was developed to give the user the output of the diesel engine simulation in a continuous manner. Using a graphical display the performance and speed of the engine simulation developed by Jones could easily be appreciated. The original graphical display consisted of a picture of a stylised engine cylinder showing a rotating piston with valves opening and closing. Moving bar graphs represented the condition of the cylinder variables: pressure, temperature, fuel air ratio and trapped mass. The graphical display also displayed a large number of numerical variables describing the state of the simulation.

As the engine simulation was developed and the simulation speed was increased this style of graphical display ceased to be useful. Information was produced from the simulation at too fast a rate to be comprehended using bar graphs and numerical displays. Also the complexity of the display meant that the picture was updated at too slow a rate. Thus as the simulation speed increased an increasingly jerky animated picture was obtained.

It is important to display the values from the simulation as they are calculated rather than displaying averaged values from the simulation. This is because the use of averaged values will lose the impression of simulation speed conveyed by the graphical display. With the increased speed of diesel engine simulation the style of any new graphical display has to be both informative and fast. A display based on the trace from an oscilloscope has therefore been developed. This gives the user a perception of the processing speed of the simulation and a time history of the displayed variables. A refinement to the oscilloscope type of display has been made. The display show the results from the engine simulation for the present engine cycle and the previous engine cycle in a different colours. This enables the user to appreciate the changes in the output from the simulation on a cycle by cycle basis. The use of the graphical display for the display of numerical data can be stopped. Numerical data can instead be displayed on the user terminal using the new menu based user interface task. This reduces the amount of information that needs to be displayed.

The next section describes the development of RUNDISP2, an animated picture design and runtime environment for the EFCIS graphics card. RUNDISP2 was used for the development of a set of new graphical displays for the diesel engine simulation. The subsequent section outlines the actual implementation of the fast animated oscilloscope style displays using RUNDISP2.

5.3.2. The development of RUNDISP2, an animated picture building environment.

Pictures are drawn on the EFCIS graphics screen by sending TRIPOS packets to the EFCIS graphics device. The type of packet actions that the EFCIS graphics device handles are given in table 5.1. The EFCIS card has two display screens,

one of which is in the foreground and is displayed and the other which is in the background. By updating the background page and then swapping the two display pages a smooth animated picture can be achieved.

The animated display developed by Jones could not easily be developed or expanded. Therefore the new animated display had to be completely redesigned. Dale [23] in his work on real time simulation of power systems developed a real time graphics environment called RUNDISP. This was designed for the EFCIS graphics card under TRIPOS. RUNDISP was used as the basis for the new graphical display for the diesel engine simulation.

RUNDISP gives a basic framework for the development of animated pictures and a runtime environment for displaying them. RUNDISP allows the creation of multiple pictures with each of the pictures being selectable by the user during the running of the program. Each picture in RUNDISP is based on five linked lists and these are as follows:

- A list of EFCIS graphics card packets representing the fixed information in the picture.
- Two lists of EFCIS graphics card packets representing the moving information in the picture. One list for each EFCIS display screen.
- Two lists of work routines. The work routines operate on the two lists of moving information packets to create a moving image.

When a RUNDISP picture is selected both of the EFCIS display pages are cleared and the fixed information packet list is written to both EFCIS display pages. RUNDISP then alternately updates each background display page with the moving picture information. First each packet in the appropriate moving picture packet list is set to the colour black and is then sent to the EFCIS graphics card. This

clears the previous moving picture information from the background display page. The picture data collection routine is then called and this fills the picture data vector with data from the simulation. The list of moving packets is then operated on by the appropriate list of work routines. The work routines use the picture data vector to update the moving packets. The updated moving picture list is then written to the EFCIS graphics card background display page. The EFCIS background page is then swapped to the foreground and displayed on the screen.

RUNDISP provides a very good animated display environment. However a number of improvements were made to RUNDISP for this research to create RUNDISP2. RUNDISP only allows two forms of picture information: fixed and moving. It is possible to draw any type of picture using these two types. However for some forms of picture this is inefficient. More EFCIS packets need to be sent to the EFCIS graphics driver than are absolutely necessary. To improve the efficiency of picture drawing and therefore the speed of drawing two other forms of picture information were identified:

- Picture information that is drawn every time that the RUNDISP picture is updated. This information is however never undrawn except by overwriting by other picture information.
- Picture information that is only drawn when certain conditions are met.

To implement these types of picture information in RUNDISP two additional picture lists were created:

- A draw only EFCIS graphics driver packet list.
- A draw conditional EFCIS graphics driver packet list.

For the conditional draw list each packet is tested to see whether it should be drawn. The packet condition flag is made part of each EFCIS packet. The second result field of the EFCIS graphics packet is unused and this can be made the condition location. As information is displayed on two pages it is important that conditional information is displayed on both graphics display pages. To achieve this the condition location is a numerical count. For each updating of a display page the condition count is decremented by one until zero is reached when the packet is no longer displayed. When a conditional packet has to be displayed the condition count is set to two. The conditional packet will then be sent to both display pages.

To make the control of the animated pictures more flexible a new packet handling control structure has been added to RUNDISP2. RUNDISP can only handle two types of control packets: picture selection packets and RUNDISP end packets. A third type of control packet type, the picture information packet has been added to RUNDISP2. The purpose of this packet is user specifiable. On receiving an information packet RUNDISP2 calls a user specified handling routine which is specified in the picture structure. The addition of this control structure makes RUNDISP2 very flexible. For example it allows the use of a standard picture layout for the display of several variables. The information packet can be used to change the picture data collection routine used by a particular picture. Thus the variable displayed can be changed. In RUNDISP the display of different variables is only possible with a separate picture for each variable that is displayed.

A pseudo code description of the operation of RUNDISP2 is given in figure 5.3. A diagram illustrating the picture list structures of RUNDISP2 is given in figure 5.4.

5.3.3. The oscilloscope type displays for the diesel engine simulation.

The development of RUNDISP2 was largely influenced by the need to make the oscilloscope displays as efficient as possible. The simplest method of implementing the oscilloscope type of display is to use two EFCIS polyvec type graphics packets in the RUNDISP moving picture packet lists. One packet is used for the display of the present engine cycle and the other is used for the display of the previous engine cycle. However this is inefficient as a large amount of static information needs to be re-sent to the EFCIS driver every time that a display page is redrawn. The picture packet for the previous engine cycle only needs to be sent once at the beginning of each new engine cycle. Similarly the picture clear packet for the engine cycle prior to the previous cycle only needs to be sent at the end of each engine cycle. Both of these can be achieved using the new RUNDISP2 conditional draw packet list.

The oscilloscope trace of the current engine cycle can also be displayed more efficiently. On each redrawing of the display pages the only thing that needs to be added to the display page is the new information that has been collected since the last updating of that display page. This can be achieved using a two point polyvec packet in the RUNDISP2 draw only packet list. On each updating of the EFCIS display page a new set of values is obtained from the engine simulation by RUNDISP2. The only new information that needs to be drawn is this new set of simulation values and the previous set of values which were only written to the other display page. The position of the polyvec packet in the draw only list is changed by the work routines in the RUNDISP2 work lists. As the extra picture information is obtained the draw only packet moves and leaves its trace behind thus giving an oscilloscope style display. The trace is unwritten by the overwriting by the previous cycle conditional packets on the next two

engine cycles.

The use of RUNDISP2 with its improved picture drawing structures reduces the number of packets sent to the EFCIS device. It therefore reduces the time spent drawing by the EFCIS card. The animated display can therefore be drawn quicker and information from the simulation can be shown with greater resolution. The reduction in the amount of information sent to the EFCIS graphics card also reduces the amount of computer backplane usage. The EFCIS graphics card is driven through the computer backplane which is also used for all the inter-processor communication of the parallel computer.

Three types of oscilloscope pictures were developed for the parallel diesel engine simulation. The first picture displayed the four cylinder variables of pressure, temperature, fuel-air ratio and trapped mass against engine crankshaft angle on four seperate oscilloscope lines. The second picture consisted of just one large oscilloscope trace for the display of just one of the cylinder variables against crankshaft angle. The last type of display was a slight variant on the oscilloscope type display and allowed the plotting of two cylinder variables against each other. For example cylinder pressure against cylinder temperature. The three types of picture are shown in figures 5.5a, 5.5b and 5.5c respectively.

5.4. The GKS graphics system for display of static display and hardcopy of engine simulation results.

RUNDISP2 is not appropriate for the static presentation of the results from the engine simulation. For this a set of graph plotting routines developed using the Graphics Kernel System (GKS) [30] are used. These provide graph plotting routines with options for scaling, grids, labelling and graph marker style. Output

from the routines can be made to the graphical screen or to plotters and printers for hardcopy of the results from the simulation.

5.5. Diesel engine simulation result organisation.

Two types of data output are taken from the diesel engine simulation. Firstly the results that are recorded on every calculation step and secondly the results that are calculated and recorded for every engine simulation cycle. The methods used for the production of both types of results differ and are detailed in the next two sections.

5.5.1. The collection and presentation of the results at each calculation step from the diesel engine simulation.

Results from the diesel engine simulation tasks are displayed immediately using the animated display described in section 5.3. For more in depth analysis of the results from the simulation the results have to be recorded for later analysis. For this each diesel engine simulation task has a result logging area in which results from the simulation task are recorded at each calculation step. The filling of the result logging area on each simulation task is started by the user interface task sending a result logging packet to that simulation task. Each simulation task then fills its result log area. When the result log area is full the result logging packet is returned to the user interface task.

The result log areas for the whole engine simulation are only useful if the results from each simulation task correspond in time. The user interface is a serial program and can only send the result logging packets to the simulation task one at a time. It is therefore possible for result logging to start at

different points on different simulation tasks. In the simulation developed by Jones [8] the result logging was controlled by the central synchronisation task which was also used to control the whole simulation. In the new parallel diesel engine simulation developed for this thesis there is no such central coordinating task. Therefore to ensure that result logging starts at the same place the engine simulation has to be halted. This ensures that all the simulation tasks are at the same place when they receive their result logging packets. The simulation can then be restarted after all the result logging packets have been sent to the simulation tasks. The simulation is halted by incorporating a halt mechanism into the engine dynamics simulation task. By halting the dynamics task the whole simulation is stopped as all the tasks in the simulation communicate with the dynamics task at each calculation step.

The following engine simulation variables are logged in the new diesel engine simulation:

1. Cylinder control volume pressure.
2. Cylinder control volume temperature.
3. Cylinder control volume fuel-air ratio.
4. Cylinder control volume trapped mass.
5. Cylinder control volume crankshaft angle.
6. Cylinder control volume inlet valve flow.
7. Cylinder control volume exhaust valve flow.
8. Manifold control volume pressure.
9. Manifold control volume temperature.
10. Manifold control volume fuel-air ratio.
11. Manifold control volume trapped mass.
12. Inlet manifold compressor inlet flow.
13. Exhaust manifold turbine outlet flow.

14. Dynamics task, simulation time.
15. Dynamics task, engine speed.
16. Dynamics task, turbine speed.

The user interface task can either display these logged simulation variables immediately, using the GKS plotting routines, or can store the results in GKS plotting format files. These GKS plotting files can be plotted later on the graphics screen or on a line plotter. An example of the GKS screen outputs are given in figure 5.6.

5.5.2. The collection and the presentation of the engine cycle results from the diesel engine simulation.

The results taken at each calculation step give a very detailed description of the state of the simulation. The derivation from the control volume variables of engine cycle variables gives more general engine performance indicators. These engine cycle variables may be compared with the results from a real engine.

The cycle parameters are calculated by each diesel engine simulation task. The engine cycle values are derived in the following three ways:

1. The average value of the simulation variables over the engine cycle.
2. The maximum value of simulation variables over the engine cycle.
3. The accumulated total of simulation variables over the engine cycle.

Each cylinder control volume task calculates the following cycle information. Where necessary the derivation of the cycle variable is given.

1. Maximum cylinder pressure. (P_{max})
2. Mass flow through the cylinder inlet valve per cycle

$$m_{iv} = \sum \frac{dm_{iv}}{d\theta} \Delta\theta \qquad \text{Equation 5.1}$$

3. Mass flow through the cylinder exhaust valve per cycle

$$m_{ev} = \sum \frac{dm_{ev}}{d\theta} \Delta\theta \qquad \text{Equation 5.2}$$

4. Mass of fuel injected into the cylinder. (m_f)

5. Energy flow through the cylinder inlet valve.

$$Q_{iv} = \sum h_{iv} \frac{dm_{iv}}{d\theta} \Delta\theta \qquad \text{Equation 5.3}$$

6. Energy flow through the cylinder exhaust valve.

$$Q_{ev} = \sum h_{ev} \frac{dm_{ev}}{d\theta} \Delta\theta \qquad \text{Equation 5.4}$$

7. Energy flow through the cylinder walls.

$$Q_{wall} = \sum \frac{dQ_{wall}}{d\theta} \Delta\theta \qquad \text{Equation 5.5}$$

8. Closed cycle work output from the cylinder

$$W_{cc} = \sum_{ivc}^{evo} (P_{cyl} - P_{amb}) \frac{dV_{cyl}}{d\theta} \Delta\theta \qquad \text{Equation 5.6}$$

9. Open cycle work output from the cylinder.

$$W_{oc} = \sum_{evo}^{ivc} (P_{cyl} - P_{amb}) \frac{dV_{cyl}}{d\theta} \Delta\theta \qquad \text{Equation 5.7}$$

10. Friction mean effective pressure (FMEP).

$$FMEP = 0.137 + 0.005P_{max} + 0.162v_{pis} \qquad \text{Equation 5.8}$$

11. Average engine crankshaft speed.

12. Volumetric efficiency.

$$\eta_{vol} = \left(\frac{P_{cyl} T_{iman} V_{cyl}}{P_{iman} T_{cyl} V_{swept}} \right)_{ivc} \qquad \text{Equation 5.9}$$

Each manifold control volume task calculates the followig cycle information.

1. Average manifold pressure.

2. Average manifold temperature.

3. Mass flow through the compressor or turbine.

$$m_{tc} = \sum \frac{dm_{tc}}{dt} \Delta t \qquad \text{Equation 5.10}$$

4. Energy flow though the compressor or turbine.

$$Q_{tc} = \sum h_{tc} \frac{dm_{tc}}{dt} \Delta t \qquad \text{Equation 5.11}$$

5. Work output from the turbine or compressor.

$$W_{tc} = \sum T_{tc} \omega_{tc} \qquad \text{Equation 5.12}$$

6. Average compressor efficiency.

7. Average turbine/compressor speed.

From these cycle variables other cycle variables may be derived and these are listed below:

1. Indicated mean effective pressure. (IMEP)

$$IMEP = \frac{W_{cc} + W_{oc}}{V_{swept}}$$

Equation 5.13

2. Pumping mean effective pressure. (PMEP)

$$PMEP = \frac{W_{oc}}{V_{swept}}$$

Equation 5.14

3. Brake mean effective pressure. (BMEP)

$$BMEP = IMEP - FMEP$$

Equation 5.15

4. Mechanical efficiency.

$$\eta_{mech} = \frac{BMEP}{FMEP}$$

Equation 5.16

5. Indicated power.

$$IP = \frac{(W_{cc} + W_{oc})\,\omega_{eng}}{4\pi}$$

Equation 5.17

6. Indicated specific fuel consumption. (ISFC)

$$ISFC = \frac{m_{fuel}}{W_{cc} + W_{oc}}$$

Equation 5.18

7. Brake specific fuel consumption. (BSFC)

$$BSFC = \frac{ISFC}{\eta_{mech}}$$ Equation 5.19

8. Indicated efficiency.

$$\eta_{ind} = \frac{W_{cc} + W_{oc}}{m_f \cdot calv}$$ Equation 5.20

9. Brake efficiency.

$$\eta_{brake} = \eta_{ind}\eta_{mech}$$ Equation 5.21

10. Delivery ratio.

$$DR = \frac{m_{iv}}{\frac{V_{swept}P_{amb}}{R_{amb}T_{amb}}}$$ Equation 5.22

11. Work output.

$$W_{out} = (W_{cc} + W_{oc})\,\eta_{mech}$$ Equation 5.23

12. Friction work output.

$$W_{fric} = W_{cc} + W_{oc} - W_{out}$$ Equation 5.24

For each of the engine control volumes the mass and the energy balances across the volumes can be used to indicate the accuracy of the results from the simulation. For the cylinder control volume the mass balance is given by equation 5.25.

$$MB = \frac{m_{iv} + m_f}{m_{ev}} \qquad \text{Equation 5.25}$$

The cylinder control volume energy balance is given in equation 5.26.

$$EB = \frac{m_f \cdot calv}{Q_{ex} - Q_{in} + Q_{wall} + W_{out} + W_{fric}} \qquad \text{Equation 5.26}$$

The mass and energy balances for the manifold control volumes are similar but they have no contribution from fuel being burnt.

The maximum simulation speed achieved by the new diesel engine simulation is about 60rpm. This means that the results from the simulation which are calculated at each engine cycle can be displayed continually on the user terminal. With an update rate of once every 2 seconds for each 720^{o} engine cycle the results are readable by the user. The user may also store the engine cycle results for later comparison with the results from a real engine.

5.6. Conclusion.

This chapter has described the development of a user friendly interface for the diesel engine. It has described in full all the methods by which the results from the parallel diesel engine are presented to the user. Using this interface the results from the engine simulation were taken for the parallel diesel engine simulation verification described in chapter 8.

TABLE 5.1. EFCIS device packet actions.

Action point. Draw a single point
argument 1 - x position
argument 2 - y position
argument 3 - colour

Action vector. Draw a vector
argument 1 - x start
argument 2 - y start
argument 3 - x end
argument 4 - y end
argument 5 - line type (solid, dotted, dashed, dot-dashed)
argument 6 - colour

Action polyvec. Draw a series of relative vectors starting at a given origin
argument 1 - x origin
argument 2 - y origin
argument 3 - buffer of x,y relative vector pairs.
argument 4 - buffer size (no. of pairs of points)
argument 5 - line type
argument 6 - colour

Action character. Draw a character at given position (x,y)
argument 1 - x position
argument 2 - y position
argument 3 - character
argument 4 - character size
argument 5 - orientation (upright, sloped, vertical, horizontal)
argument 6 - colour

Action string. Draw a string at given position (x,y)
argument 1 - x position
argument 2 - y position
argument 3 - BCPL string
argument 4 - character size
argument 5 - orientation (upright, sloped, vertical, horizontal)
argument 6 - colour

Action clear. Clears screen to given colour.
argument 1 - colour

Action colour. Sets EFCIS colour pallette.
argument 1 - colour no.
argument 2 - red magnitude
argument 3 - green magnitude
argument 4 - blue magnitude

Action flip. Sets EFCIS display and write pages.
argument 1 - bit flags (b7 = display page, b3 = write.page)

Action rmw. Sets EFCIS read modify write mode.
argument 1 - Flag (true, false)

Figure 5.1a The parallel diesel engine simulation, main menu screen.

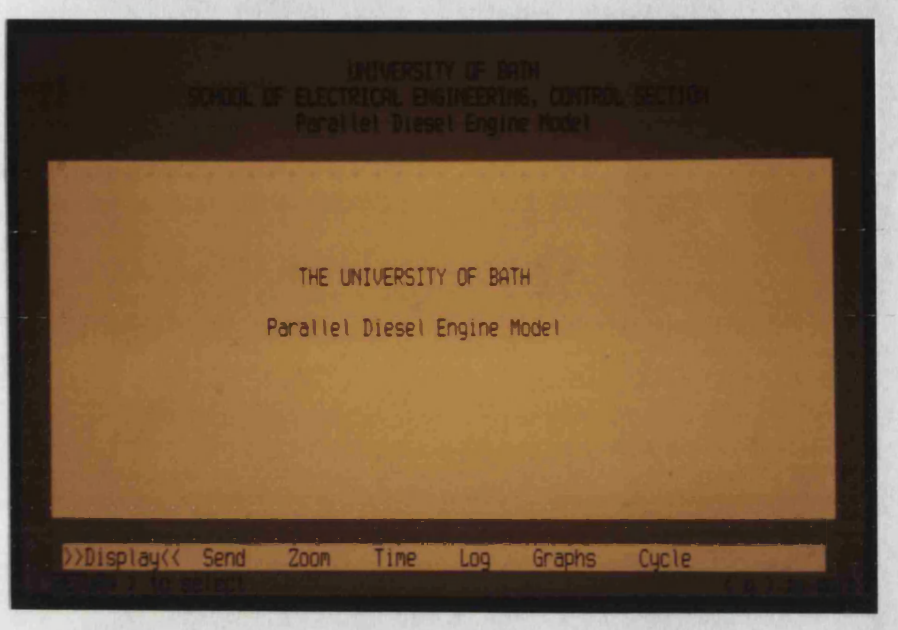


Figure 5.1b The parallel diesel engine simulation, cycle data display screen.

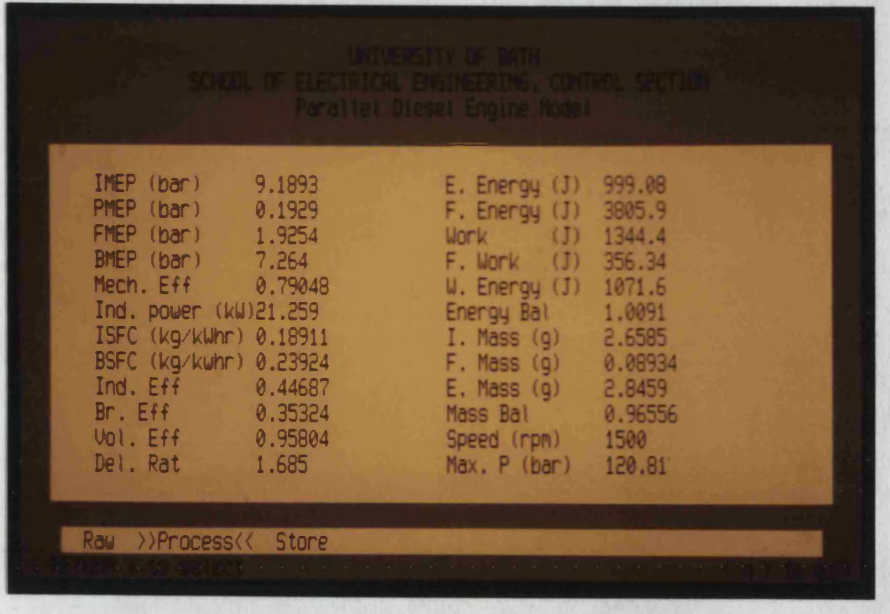
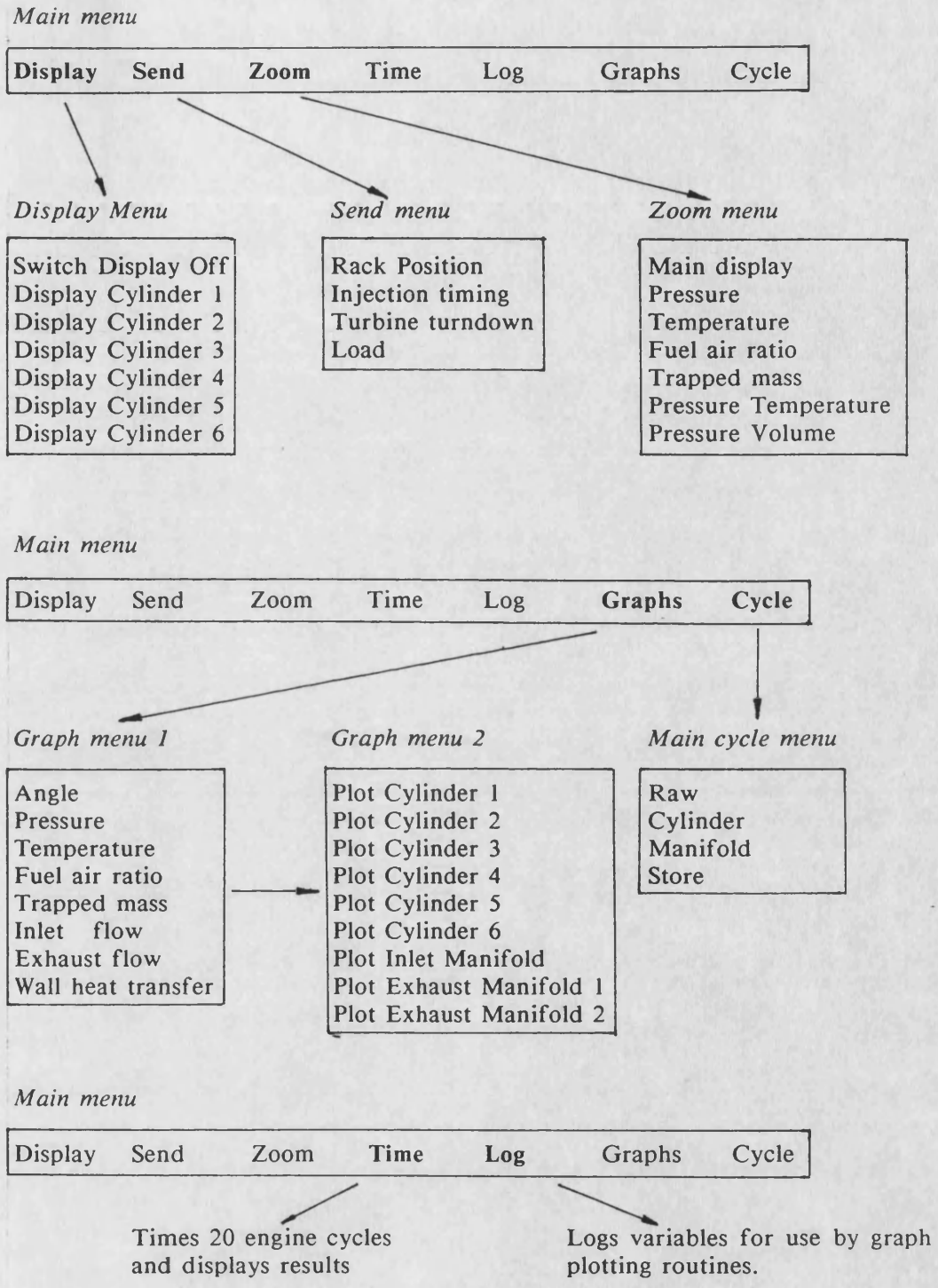


Figure 5.2 The diesel engine simulation menu interface structure.



Main cycle menu

Raw	Cylinder	Manifold	Store

Raw cycle data menu display

O. cycle work (J)	Man. pres (Pa)
C. cycle work (J)	Man. temp (K)
Inlet mass (kg)	TC. speed (rad/s)
Exhaust mass (kg)	TC. mass (kg)
Inlet energy (J)	Cyl. mass (kg)
Exh. energy (J)	TC. energy (J)
Fuel mass (kg)	Cyl. energy (J)
Max. pres. (Pa)	TC. work (J)
FMEP (Pa)	TC. eff
Wall energy (J)	
Volumetric eff.	
Eng. speed (rpm)	

Processed cylinder cycle data display

IMEP (bar)	Delivery ratio
PMEP (bar)	Exh. Energy (J)
FMEP (bar)	Fuel Energy (J)
BMEP (bar)	Work (J)
Mechanical Eff.	Friction Work (J)
Ind. power (kW)	Wall Energy (J)
Brake power (kW)	Energy Balance
ISFC (kg/kWhr)	Inlet Mass (g)
BSFC (kg/kwhr)	Fuel Mass (g)
Indicated Eff.	Exhaust Mass (g)
Brake Efficiency	Mass Balance
Volumetric Eff.	Speed (rpm)
Fuel flow (g/s)	Max. Pres. (bar)

Processed manifold cycle data display

Turbine speed (rpm)	Compressor Eff.
In. man pres (bar)	In. man temp (K)
Ex1 man pres (bar)	Ex1 man temp (K)
Ex2 man pres (bar)	Ex2 man temp (K)
In. man mass in (g)	In. man mass o. (g)
Ex1 man mass in (g)	Ex1 man mass o. (g)
Ex2 man mass in (g)	Ex2 man mass o. (g)
In. man E in (J)	In. man E out (J)
Ex1 man E in (J)	Ex1 man E out (J)
Ex2 man E in (J)	Ex2 man E out (J)
In. man mass bal	In. man Energy bal
Ex1 man mass bal	Ex1 man Energy bal
Ex2 man mass bal	Ex2 man Energy bal
Comp. power (kW)	Turbine1 power (kW)
	Turbine2 power (kW)

Figure 5.3. Flowchart of RUNDISP2 graphics runtime environment.

```
LET start (initial packet) BE
$(   build picture data structures specified in user file.
     current picture := 1
     initialise (current picture)
     returnpkt ( initial packet )
     $(   // The main picture display loop.
          IF picture change packet present
          THEN
          $(   current picture := packet argument 1
               UNLESS current picture = stop display DO
                    Initialise ( current picture )
               returnpkt ( picture change packet )
          $)
          TEST current picture = -1
          THEN process (current picture)
          ELSE handle.pkt ( taskwait() )
     $) REPEATUNTIL self immolate packet present
     endtask()
$)

AND initialise ( picture ) BE
$(   clear screen 1
     draw picture fixed list on screen 1
     clear screen 2
     draw picture fixed list on screen 2
$)

AND process ( picture ) BE
$(   undraw the picture moving picture packet list on background screen
     request picture data from simulation tasks
     update the picture packet lists with picture data
     draw the picture draw only packet list
     draw the picture draw conditional packet list
     redraw the picture moving picture packet list
     swap EFCIS display pages to display updated screen
$)

AND handle.pkt ( packet ) BE
$(   SWITCHON packet type field INTO
     $(   CASE action self immolate :
               mark self immolate packet present
          ENDCASE
          CASE action picture change :
               mark picture change packet present
          ENDCASE
          CASE action picture information :
               picture information routine ( packet )
               returnpkt ( packet, true, 0)
          ENDCASE
          DEFAULT :
               returnpkt ( packet, false, 0)
     $)
$)
```

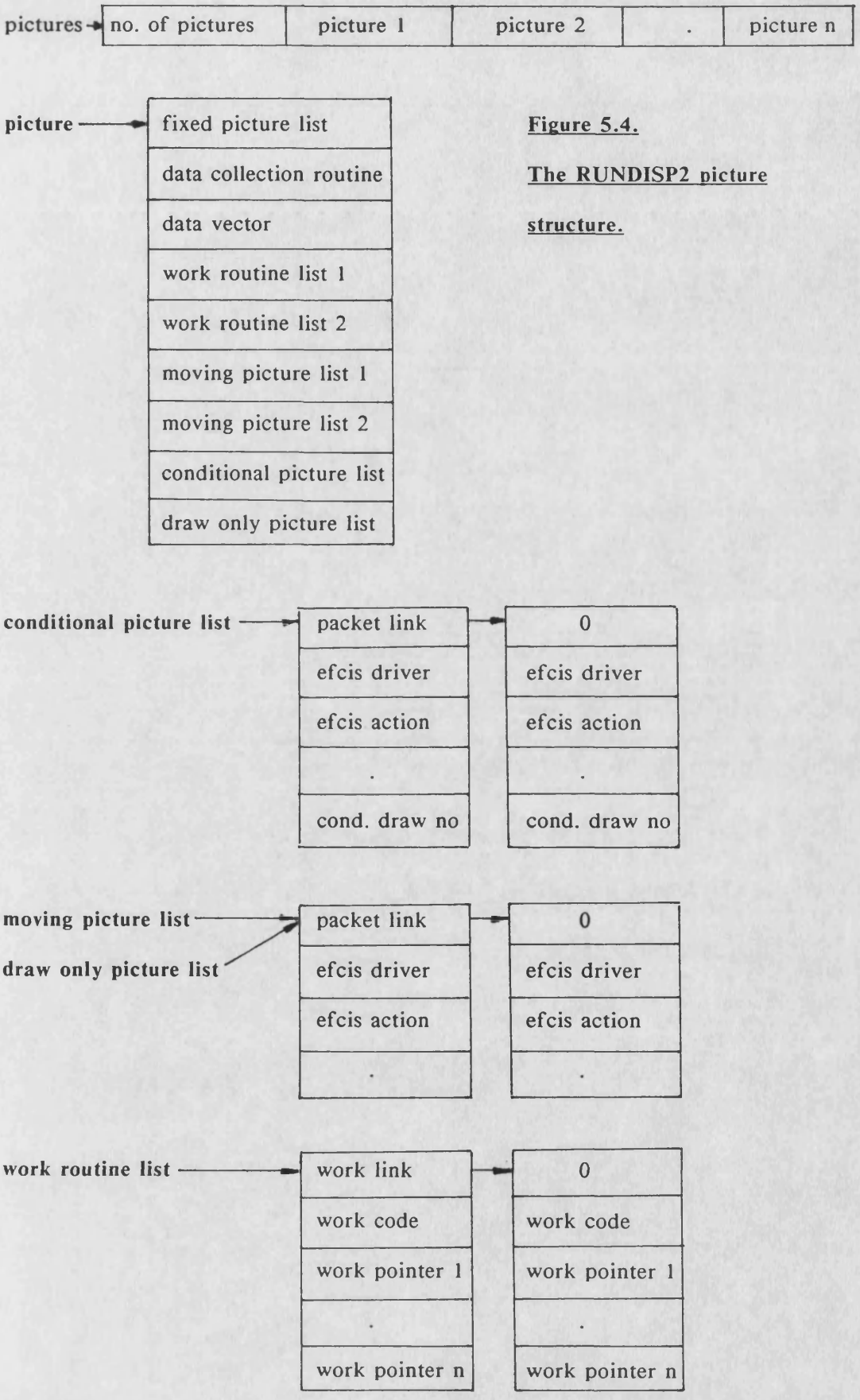


Figure 5.4.

The RUNDISP2 picture structure.

Figure 5.5a Cylinder pressure, temperature, fuel-air ratioand mass oscilloscope display.

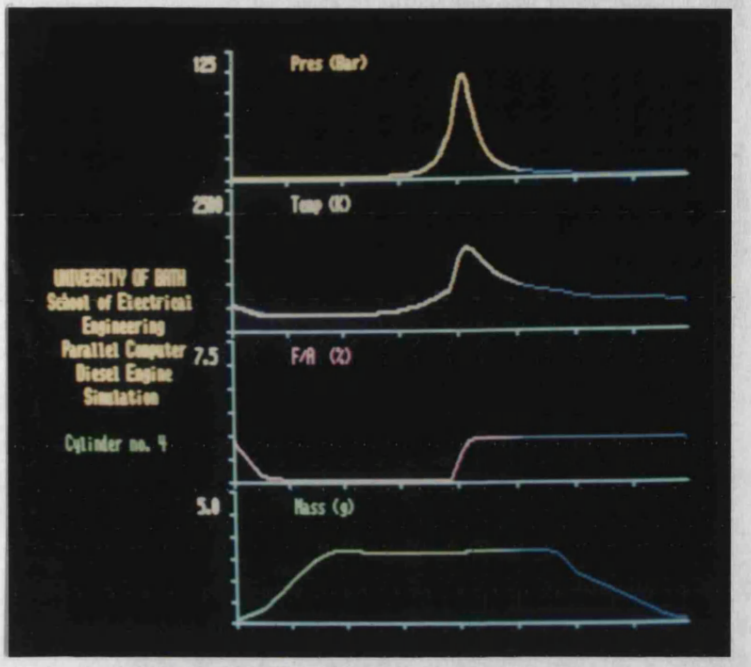


Figure 5.5b Cylinder pressure oscilloscope display.

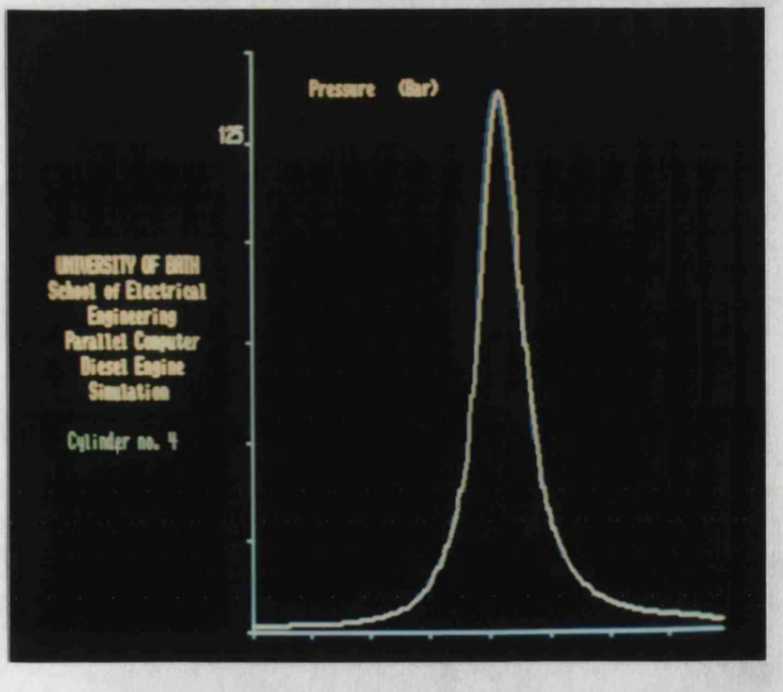


Figure 5.5c Cylinder pressure against volume oscilloscope display.

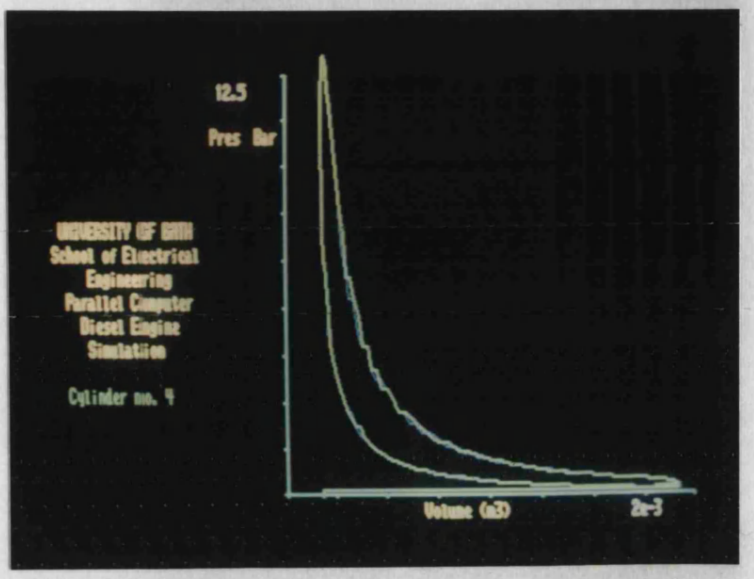
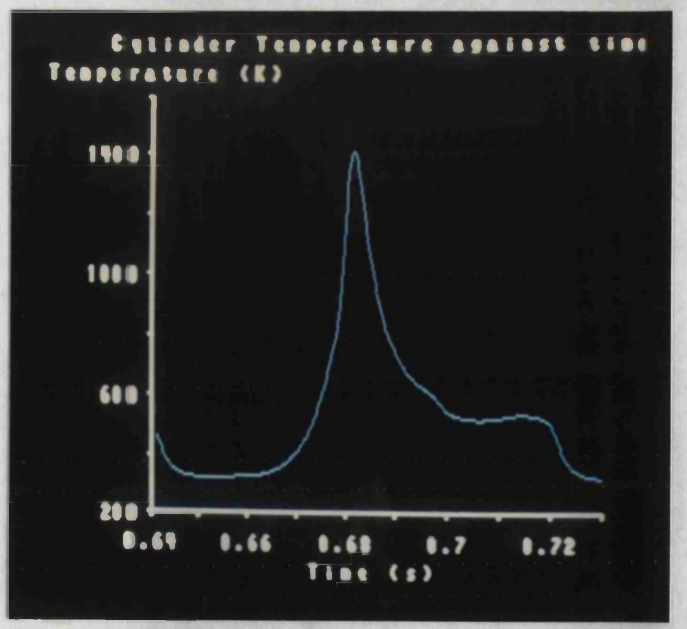


Figure 5.6 Cylinder temperature against time GKS screen display.



CHAPTER 6.

INCREASED PARALLELISM IN THE SIMULATION OF DIESEL ENGINES.

6.1. Introduction.

As discussed in Chapter 4, the Diesel engine 'filling and emptying' simulation of a multi-volume Diesel engine has been implemented on a parallel computer. The splitting of the simulation into parallel sections is based on the control volumes and dynamics functions of the engine being modelled. The parallel sections of the Diesel engine 'filling and emptying' simulation are 'geometric' in nature. 'Geometric' parallelism is where the parallelism is based on the physical structure of the object being modelled. For the Diesel engine simulation, this type of parallelism is limited to the number of control volumes and dynamic parts of the Diesel engine.

The parallel simulation described in chapter 4 has demonstrated that the use of parallelism can greatly improve the speed of Diesel engine simulation. In chapter 9 the uses for very fast Diesel engine simulation are discussed. The identification of more parallelism in the Diesel engine simulation will lead to increased simulation speed. This will be of benefit in the applications outlined in Chapter 9.

A number of methods for increasing the parallelism of the 'filling and emptying' Diesel engine simulation have been researched for this thesis. These are the following:

- The splitting of the engine control volume equations into smaller parallel sections.
- Concurrent solution of the engine control volume equations at different integration step lengths to maximise integration step length.
- Parallel integration algorithms.

They are discussed in the following sections. A number of the new parallel methods have been practically implemented32Xand the results from these are discussed.

6.2. Increased parallelism in the Diesel engine simulation through control volume equation segmentation.

The engine control volume equations consist of a number of distinct sub-equations. The calculation of many of these sub-equations is independent of the calculation of the others. They can therefore be calculated on different processors in parallel. The amount of parallelism that can be obtained by segmentation of the control volume equations may be determined by an analysis of the control volume equations. Figure 6.1 shows this equation analysis graphically. It indicates the possibilities for parallelism for the cylinder control volume equations. The parallel calculation of sub-equations is ultimately still limited by the Diesel engine being simulated. It is therefore still 'geometric' in nature.

The investigation of sub-equation parallelism is suggested by Jones [8]. However a number of problems can be identified with the increased parallelism due to control volume equation segmentation. These are described in the following sections.

6.2.1. The problem of variable parallelism in the control volume sub-equations.

The parallelism obtained by the splitting of the engine control volume equations into parallel sections varies considerably through the simulation cycle of the

engine. For example, the concurrent calculation of the cylinder inlet valve flow and the cylinder exhaust valve flow is only present during the scavenge section of the cylinder engine cycle. An allocation of one processor to each sub-equation calculation would use some of the parallel processing nodes inefficiently. To overcome this processors could be used for the calculation of valve flows for several cylinders. However, it is possible for two cylinder volumes to both need the valve flow calculations at the same time for the same calculation step. This would cause one of the cylinder volumes to wait and would incur a time penalty over direct computation by one processor.

6.2.2. The problem of processor load balance.

The splitting of the control volume equations into parallel sections produces a problem of processor load balance. The parallelism based on engine control volumes used in this research has good processor load balance. The parallel sections of the Diesel engine simulation, which are based on engine control volumes, have approximately equal computational and communication needs. These parallel sections are therefore not forced to wait for a long time for the completion of other parallel sections. The sub-equations within each engine control volume can vary considerably in computational difficulty. The speed of the parallel execution will therefore be determined by the computationally most difficult parallel sub-equation section.

The problem of processor load balance also causes an increase in processing time. Under multi-processor TRIPOS a task will suspend when it has to wait for a packet to arrive from another task. This suspension and subsequent restarting take time. With a well balanced solution the amount of task suspension can be reduced.

6.2.3. The problem of increased communication to processing ratio.

In a serial program all the processing time is spent in calculation. In a parallel computer program processing time is spent both in calculation and communication with other processors. For an efficient parallel computer program the time taken in communication and calculation by each of the parallel sections should be less than the sum of the calculation times for all the parallel sections. As the size of the parallel sections is reduced and the nature of the parallel problem becomes 'finer grained' the ratio of communication time to calculation time will tend to increase. The inter-processor communication bandwidth of a parallel computer is fixed. This will limit the degree to which equations can be divided up into parallel sections. Subdivision of equations below a certain threshold will produce a processing time penalty. The increase in communication time will outweigh the reduction in calculation time due to the parallelism.

6.2.4. The practical implementation of cylinder control volume sub-equation parallelism.

The feasibility of sub-equation parallelism on the Bath University Parallel Computer (BUPC) was tested for this thesis. For this a simple single cylinder engine simulation was used. The single cylinder simulation has fixed conditions at both the inlet and exhaust valves of the cylinder control volume. It therefore has no communication with manifold tasks and so it will eliminate the effect of inter-volume communication from the test. This allows the effects of sub-equation parallelism to be identified and analysed in isolation. Results can be compared with those for a simple single processor, single cylinder simulation.

The sub-equations for calculating the cylinder volume gas properties and calculating the heat transfer to the cylinder volume surfaces may be calculated in parallel. The calculation of these does not vary with cylinder control volume sector so this parallelism is used throughout the engine cycle. The calculation of the cylinder volume heat transfer was implemented on a second processor. The main processor carried out all the cylinder control volume calculations except the calculation of heat transfer. It calculated the cylinder gas properties in parallel to the heat transfer calculation on the second processor. The main processor sent the heat transfer calculation conditions to the second processor on each equation iteration. The main processor then continued with the calculation of the gas properties before waiting for the heat transfer result from the second processor. The second processor simply waited for the calculation conditions from the main processor and returned the heat transfer result at the end of its calculation. All communication between the parallel sections used the multi-processor TRIPOS communication primitives. The parallelism of this test is shown in figure 6.2.

The use of this parallelism was found to cause an increase in program execution time over the time taken for the serial program. The communication time between the two parallel equations was greater than the time saved due to the parallelism. The problem was too 'fine grained' for the MC68020 Bath University Parallel Computer.

The one cylinder engine simulation used two processors with no other processors using the computer backplane during the test. For a larger example of parallelism where more than two processors would be used there would be an increase in communication time between the parallel sub-equation sections. This would be due to increased inter-processor communication and therefore computer backplane

access clashes. The reduction in calculation speed due to sub-equation parallelism would therefore get worse due to an increase in communication time. Table 6.1 gives the calculation times for each of the cylinder control volume sub-equations. The approximate times for the TRIPOS communication primitives are also given.

TABLE 6.1 Cylinder sub-equation timings and TRIPOS communication timings.

Calculation routine.	**Time.**
Inlet valve flow calculation.	16.58μs
Exhaust valve flow calculation.	16.5μs
Gas properties calculation.	28.9μs
Gas properties calculation. (compression)	9.62μs
Wall heat transfer calculation.(look up tables)	10.16μs
Wall heat transfer calculation.	36.4μs
Pressure calculation.	2.66μs
Temperature gradient calculation $dT/d\theta$.(scavenge)	7.02μs
Fuel-air gradient calculation $d\lambda/d\theta$. (scavenge)	6.94μs
Mass gradient calculation $dm/d\theta$. (scavenge)	1.6μs
Tripos QPKT. (to another processor)	70μs
Tripos TASKWAIT. (from another processor)	120μs

The use of sub-equation parallelism can only be considered in a parallel computer with a higher communication to processing speed ratio or where the communication primitives are more efficient than the multi-processor TRIPOS QPKT and TASKWAIT communication primitives. Consideration was given to improving the speed of inter-processor communication between the parallel sub-equation sections. This could be achieved by implementing specialist inter-

processor communication routines. However this was not fully implemented as the results would have been very specific to the MC68020 computer. The specialist synchronisation routines would not have been readily applicable to other parallel computers such as the new Transputer computer outlined in chapter 3.

6.3. Creating parallelism in the Euler predictor corrector method to achieve maximum speed through optimising integration step length.

The modified Euler predictor corrector method is a serial algorithm. It is possible to adjust the integration steplength of this algorithm as the solution proceeds. In the Diesel engine simulation based on control volume parallelism the integration steplength is halved when the integration method is unstable at the original steplength.

To achieve maximum simulation calculation speed the integration steplength should be maximised. This can however cause increased use of integration steplength reduction in areas of integration instability. This in turn leads to a decrease in engine simulation speed. In chapter 4 the speed of the multi-volume Diesel engine simulation was measured with the initial integration steplength used being varied. These results indicated that a maximum simulation speed is reached for an integration steplength of 2^{o}. Above this the amount of integration steplength reduction outweighs the speed gains obtained through using a larger integration steplength.

The use of several processors allows the same integration calculation to be solved on different processors with different integration steplengths. The solution with the maximum steplength, that gives stability, can then be used to continue the simulation. This will give the maximum integration speed for the engine

simulation.

Tests on a single cylinder engine simulation using the Euler predictor corrector method were carried out for this thesis to determine the feasibility of this method. It was found that an upper limit to integration steplength is reached. Above this steplength stability can still be achieved but with incorrect simulation results. As this parallel method would attempt to maximise integration steplength it would cause this problem of result inaccuracy. These incorrect results are due to the poor error characteristics of the Euler predictor corrector formulae at large steplengths. It might be possible to use this parallel method with a better serial integration method.

6.4. 'Algorithmic' parallelism in the calculation of ordinary differential equations.

All the parallelism discussed so far may be termed 'geometric' parallelism. The parallelism used is based on the problem being solved. Another form of parallelism which is known as 'algorithmic' parallelism is possible. For this the parallelism is implicit in the algorithm used to solve the problem.

The Diesel engine 'filling and emptying' model described in chapter 2 is a set of ordinary differential equations. The algorithm used to solve the parallel Diesel engine simulation described in Chapter 4 is the modified Euler predictor corrector algorithm. This is a serial algorithm. Several parallel algorithms for solving ordinary differential equations have been researched for this thesis. Two of these methods have been practically implemented for the Diesel engine 'filling and emptying' simulation equations. The parallel integration methods which have been researched are described in the next four sections.

6.4.1. The Miranker and Liniger integration method.

Miranker and Liniger [46] outline a parallel integration method for solving differential equations with concurrent prediction and correction of different solution points. They describe a standard predictor corrector algorithm as having a 'narrow computation front' which is unable to take advantage of parallel computing. A series of parallel integration formulae for an even number of processors are derived by Miranker and Liniger. In these formulae the 'computation front' is widened to allow parallel processing. An example of the second order Miranker and Liniger formulae for two processors is given in equations 6.1.

$$y^p_{n+1} = y^c_{n-1} + 2hf^p_n \qquad \text{Equation 6.1a.}$$

$$y^c_n = y^c_{n-1} + \frac{h}{2} \cdot (f^p_n + f^c_{n-1}) \qquad \text{Equation 6.1b.}$$

This is similar in computational difficulty to the serial modified Euler predictor corrector algorithm equations 6.2 which has been used for the Diesel engine simulation in chapter 4.

$$y^p_{n+1} = y^c_n + hf^p_n \qquad \text{Equation 6.2a.}$$

$$y^c_{n+1} = y^c_n + \frac{h}{2}(f^p_n + f^p_{n+1}) \qquad \text{Equation 6.2b.}$$

Figures 6.3a and 6.3b illustrate the concept of the broadening of the 'computation front' to allow parallel processing. The computation front is shown by the dashed line. The figures 6.3a and 6.3b show the data flow for equations 6.1 and 6.2 respectively. The Miranker and Liniger integration method shows perfect load balance between its parallel sections. This is shown in figure 6.4 where the sequence of computation for equation 6.1 is given.

The Miranker and Liniger integration method does not introduce any computational redundancy into the integration equations to achieve parallelism. It is however difficult to adjust the integration steplength of the method during the solution of the differential equations. The adjustment of the integration steplength would allow the method to take account of areas of integration instability. Kerkchoff [47] notes that the Miranker and Liniger integration method has the same error characteristics as the equivalent serial integration method. However he also notes that the stability of the Miranker and Liniger integration method is considerably smaller.

6.4.2. The one step block implicit integration method.

A standard integration algorithm removes all redundant computation to achieve maximum efficiency. However by introducing redundancy into standard methods these methods may be reframed in a way suitable for parallel solution. Shampine and Watts [48] derive a set of implicit integration formulae for parallel integration. These are known as block implicit one step methods. The formulae are of a Newton-Cotes form. For every pass of the equations k new equally spaced solutions are produced. The equations are applied in a one step mode where the last point calculated in the previous block is used to predict the k values in the next block. The formulae for the four point one step block implicit integration method are given in equations 6.3.

$$
\begin{aligned}
y_{n+r,0} &= y_n + r \cdot h \cdot f_n \\
y_{n+1,s+1} &= y_n + \frac{h}{720}(251f_n + 646f_{n+1,s} - 264f_{n+2,s} + 106f_{n+3,s} - 19f_{n+4,s}) \\
y_{n+2,s+1} &= y_n + \frac{h}{90}(29f_n + 124f_{n+1,s} + 24f_{n+2,s} + 4f_{n+3,s} - f_{n+4,s}) \\
y_{n+3,s+1} &= y_n + \frac{3h}{80}(9f_n + 34f_{n+1,s} + 24f_{n+2,s} + 14f_{n+3,s} - f_{n+4,s}) \\
y_{n+4,s+1} &= y_n + \frac{2h}{45}(7f_n + 32f_{n+1,s} + 12f_{n+2,s} + 32f_{n+3,s} + 7f_{n+4,s})
\end{aligned}
$$

Equations 6.3.

Rosser [49] shows that the one step block integration method has computational redundancy. He shows how the number of derivative evaluations can be reduced to give a serial one step block integration method. This serial integration method is applied in a Gauss-Seidel sense with each equation evaluation being used in the subsequent evaluation. The parallel integration method is applied in a Jacobi type recursion where each equation evaluation is based on the equation evaluations from the previous step.

A comparison of the number of function evaluations for the four point serial and parallel one step integration methods gives fourteen and twenty respectively. However for the parallel integration method four processors may be used. This gives an equivalent number of function evaluations on each processor of five. If the function evaluation time dominates the calculation time, which it does in the Diesel engine model, then the parallel integration method gives a theoretical speed up of 2.8 over the serial block integration method. The stability of the serial and the parallel block integration methods is similar. Kerkchoff [47] indicates that for a four point integration method with four iterations of the formulae gives the same characteristics as a sixth order Runge-Kutta formula.

6.4.3. The block predictor corrector integration method.

Birta and Abou-Raita [50], Rosser [49], and Worland [51] derive equations for a full block predictor corrector form of the one step formulae described in section 6.4.2. They derive a set of explicit block equations for use as a block predictor. In these all the solution values in the previously calculated block are used to predict the solutions in the next block. The formulae for the four point block predictor corrector method are given in equations 6.4.

$$
\begin{aligned}
y_{n+1,0} &= y_n + \frac{h}{720}(1901f_n - 2774f_{n-1,1} + 2616f_{n-2,1} - 1274f_{n-3,1} + 251f_{n-4,1}) \\
y_{n+2,0} &= y_n + \frac{h}{90}(1079f_n - 2396f_{n-1,1} + 2544f_{n-2,1} + -1316f_{n-3,1} + 269f_{n-4,1}) \\
y_{n+3,0} &= y_n + \frac{h}{80}(2877f_n - 7638f_{n-1,1} + 8712f_{n-2,1} - 4698f_{n-3,1} + 987f_{n-4,1}) \\
y_{n+4,0} &= y_n + \frac{h}{45}(3914f_n - 11456f_{n-1,1} + 13704f_{n-2,1} - 7616f_{n-3,1} + 1634f_{n-4,1}) \\
y_{n+1,1} &= y_n + \frac{h}{720}(251f_n + 646f_{n+1,s} - 264f_{n+2,s} + 106f_{n+3,s} - 19f_{n+4,s}) \\
y_{n+2,1} &= y_n + \frac{h}{90}(29f_n + 124f_{n+1,s} + 24f_{n+2,s} + 4f_{n+3,s} - f_{n+4,s}) \\
y_{n+3,1} &= y_n + \frac{3h}{80}(9f_n + 34f_{n+1,s} + 24f_{n+2,s} + 14f_{n+3,s} - f_{n+4,s}) \\
y_{n+4,1} &= y_n + \frac{2h}{45}(7f_n + 32f_{n+1,s} + 12f_{n+2,s} + 32f_{n+3,s} + 7f_{n+4,s})
\end{aligned}
$$

Equations 6.4.

Birta and Abou-Raita [50] derive an algorithm for the optimum adjustment of the integration steplength size in the block predictor corrector integration method. This is based on an analysis of the error characteristics for the previous block. The maximum error for all the points in the block is taken as the error for the whole block. The error for each point in the block is a normalised error derived from the difference of the predicted and corrected values and a permissible error value. The error calculation is given in equation 6.5.

$$R = \left| \frac{(y_i - y^p{}_i)}{\epsilon} \right|$$

Equation 6.5.

The block maximum error is used as the basis for a steplength adjustment scheme. The size of the steplength adjustment is given in equations 6.6 and is dependent on the point in the block from which the maximum error was derived and the number of points in the block.

$$\begin{aligned} h_{new} &= \sigma h_{old} \\ \sigma &= \left(\frac{1}{R}\right)^{\gamma} \\ \gamma &= \frac{1}{\rho+1} \end{aligned}$$

Equations 6.6

The block maximum error determines the progression of the solution. If the error is less than unity then the solution may proceed. If the error is greater than unity then the block step is repeated. In both cases the block is calculated using the new steplength derived from the steplength adjustment given in equation 6.6.

6.4.4. The power series integration method.

From a consideration of the solution of ordinary differential equation by means of a recursive Taylor series integration method, a semi-analytical parallel integration method may be derived. The Taylor series is given in equation 6.7.

$$y_{n+1} = y_n + h\dot{y} + \frac{h^2}{2!}\ddot{y} + \ldots \frac{h^k}{k!}y^{(k)} \qquad \text{Equation 6.7}$$

This method involves the decomposition of the ordinary differential equations into simple two operand arithmetic equations such as *a+b* or single operand functions such as *sin(x)*. For example equation 6.8(1) may be decomposed to give equations 6.8(2)-(7).

$$\frac{dy}{dt} = ye^{ay} + \sin(y+b) \quad (1)$$
$$p = ay \quad (2)$$
$$q = e^{p} \quad (3)$$
$$r = yq \quad (4)$$
$$s = y + b \quad (5)$$
$$u = \sin s \quad (6)$$
$$\frac{dy}{dt} = r + u \quad (7)$$

Equations 6.8

Equations relating the kth order derivative of a variable to lower derivatives of that variable and others can be derived. Equation 6.9 outlines the calculation needed for the kth derivative of the function $p.q$ where p and q are variables.

$$r = p \cdot q$$
$$r^{(v)} = \sum_{s=0}^{v} p^{(s)} \cdot q^{(v-s)}$$

Equation 6.9

Using these derivative formulae the solution of the differential equations becomes the iterative application of a set of scalar product equations and their summation. On each iteration the next highest order derivative for each operand is added to the solution. The iteration is stopped when a stability criterion is reached where the difference between the last two iteration results is below a defined stability threshold.

The parallelism of the power series method is due to the ability to solve the power series of the decomposed functions and arithmetic operations in parallel. The power series integration method has been used by Halin et al. [52] on a multi-processor computer. It is felt to offer a general purpose solution to simulation on a parallel computer. The decomposition of simulation equations to

suit the integration method could be automated giving ease of use. However the power series integration method produces a very 'fine grained' parallelism which is felt to be inappropriate to the present Bath University Parallel Computer. This is because of the limited number of processing nodes of the computer and its inter-processor communication speed. The Bath University Parallel Computer is more suited to 'coarse grained' parallelism.

6.5. The implementation of the block predictor corrector integration method.

It is important that the parallel integration method chosen for solving the Diesel engine model should be capable of coping with changing the integration steplength. This is considered important as the complexity of the Diesel engine model equations varies considerably through the engine cycle. The ability to reduce the integration steplength in parts of the engine cycle such as scavenge where the equations are more difficult to solve is an advantage. It allows the use of an optimum steplength throughout the simulation. The block predictor corrector integration method was therefore used as it allows steplength adjustment. The block predictor corrector integration method also allows the use of iteration when applying the correction formulae to obtain a more stable solution. The Miranker and Liniger integration method [46] only considers the application of the corrector formulae once. Use of the corrector formulae more than once in the Miranker and Liniger integration method would lose the parallelism of this integration method.

Franklin [53] indicates that the one step block predictor corrector integration method has a higher stability boundary than the Miranker and Liniger integration method. He also states that computation times for similar order integration equations are similar for both the block predictor corrector integration method

and the Miranker and Liniger integration method.

Birta and Abou-Raita [50] discuss the likely speed up in processing speed from using the block predictor corrector integration method compared to a similar method on a serial processor. They suggest that any speed up from using two parallel processors is cancelled out by the increased communication between the processors. A block size of four was therefore chosen. With four processors a speed up in equation solution is achievable. It was also implementable on the six processor parallel computer which was then being used for the research in this thesis.

6.5.1. The initial implementation of the block predictor corrector integration method using simple equations.

The block predictor corrector integration method was initially implemented using a simple ordinary differential equations with analytic solutions. The equation and their solutions are given in equations 6.10.

$$\begin{aligned} \frac{dy}{dt} &= \cos t \\ y(0) &= 1 \\ y &= \sin t + 1 \end{aligned}$$

Equations 6.10

The use of these simple equations allowed the problems associated with the block predictor corrector integration method to be analysed in isolation.

6.5.2. The problem of inter-processor synchronisation speed.

A four point block predictor corrector integration method requires the exchange of four calculated function values from each processor on each iteration of the prediction and correction formulae. Initially this was achieved using the multi-processor TRIPOS communication primitives of QPKT and TASKWAIT. However it was found that synchronisation using these was too slow and resulted in a slower calculation speed than the single processor modified Euler predictor corrector formulae.

6.5.3. Specialist synchronisation routines and the communication of data for the block predictor corrector integration method.

The four calculated function values on each processor can be transferred more quickly by direct reading of the values by each processor than by using the TRIPOS communication primitives. This requires the synchronisation of the four processors prior to the reading of the function values on the other processors. This synchronisation ensures that the function values are valid and may be read by the other processors. The synchronisation may be achieved using the TRIPOS communication primitives but this is as slow as communication with them. The four processors involved in the block predictor corrector algorithm can be considered as a cohesive unit or quartet. The writing of specialist synchronisation routines gives considerable speed advantages for the synchronisation of such a quartet. By using the TRIPOS communication primitives for all other communication in the system the general purpose nature of communication can be maintained.

In the design of the processor synchronisation routines care has to be taken with common information used by more than one processor. In the Bath University Parallel Computer, which is a shared memory computer, the memory locations used by more than one processor have to be protected. This protection is to ensure that the consistency of the values in the memory is maintained. As all the processors are asynchronous no assumptions about the timing of processor actions can be made. Actions on shared areas of memory therefore must be made indivisible. A processor cannot be allowed to use a memory location between another processor starting and completing an action on that same memory location.

Three alternatives for synchronisation of processors for the block predictor corrector integration method were considered. These are discussed in the following three sections.

6.5.3.1. Processor synchronisation using semaphores.

The use of semaphores for multi-process communication was introduced by Dijkstra [54]. A semaphore is a non-negative integer which may be acted upon by two indivisible operations signal and wait. A signal has the effect of increasing the value of a semaphore by one. The wait decreases the semaphore value by one if the result will be non negative. If the semaphore is at zero then a wait will suspend the waiting process until the semaphore value is increased by a signal from another process. Semaphores were originally developed for concurrent processes in single processor operating systems. They are equally applicable to multi-processor systems.

In a single processor system the indivisible access to a semaphore may be

ensured by using a simple flag to gain access to the semaphore. In a multi-processor system the testing of this flag and its setting must also be made indivisible. For the Motorola MC680XX processors [25] used in the Bath University Parallel Computer this can be achieved with the indivisible read-modify-write Test and Set instruction (TAS).

The block predictor corrector processor quartet can be synchronised with semaphores. Each member of the processor quartet has a local semaphore. This local processor semaphore is signalled by all other processors in the quartet when they reach the synchronisation point. Having signalled the other processor semaphores each processor then executes three waits on its local semaphore before proceeding. Each wait is for each synchronising processor.

The signalling of the semaphores by each processor occurs sequentially. It is therefore possible for one processor to have its semaphore signalled three times and to continue from the synchronisation point to the next synchronisation point. The advanced processor can then signal the synchronisation semaphores again before other processors have signalled the semaphores for the first synchronisation. The signalling of the semaphores by the advanced processor can be misinterpreted by the other processors. However the fact that the advanced processor has been synchronised ensures that all data on all of the processors in the quarter is valid for the next calculation step. The misinterpretation of any signals from the advanced processor as coming from the other processors will therefore not matter. The total number of signals will not be lost and so processor 'deadlock' will not occur.

6.5.3.2. Processor synchronisation by waiting on local flags.

An alternative method to the use of semaphores is to use simple flags on each of the processors to synchronise the processor quartet. For this synchronisation method each synchronising processor has a separate flag in its local memory for each of the synchronising processors. On every synchronisation each processor sets its respective flag on all the other processors. Each processor is then made to wait for all its local flags to be set by the other processors. Every processor may then clear its local flags and continue.

For this simple scheme it is possible for a processor to continue and reach the next synchronisation point prior to another processor clearing its flag for the first synchronisation. In this case it is possible for the first processor to set its synchronisation flag on the second processor for a second time. This can occur before the second processor has cleared this flag. This second setting of the flag will therefore be lost as the flag can only be set or cleared. A processor 'deadlock' will therefore occur for the next synchronisation.

The double setting of the processor flags can be prevented by implementing two identical synchronisation routines. Each synchronisation routine has a unique set of processor flags. The two routines can then be used alternatively with one protecting the other from the double setting of its flags. The clearing and the setting of the flag locations is effectively double buffered.

6.5.3.3. Processor synchronisation by waiting on remote flags.

The opposite of the above method is for each synchronising processor to have its own local flag. This flag is set by the local processor for each synchronisation.

The processors in the quartet synchronise by testing each of the flags on all the other processors and waiting until they are all set. The flags may then be cleared.

This method is difficult to implement because of the problem of organising the clearance of the synchronisation flags. A method that ensures that processor 'deadlock' does not occur requires a set of three synchronisation routines, the three routines being used in turn. Each synchronisation routine is responsible for the clearing of the local processor flag used in the previous synchronisation routine. The use of two routines as in the previous method is not possible. Clearance of the previous flag after the second synchronisation would affect the flag to be used in the subsequent synchronisation.

As an alternative to the use of just one local flag, each processor can maintain a local flag for each processor. Each synchronising processor can then be made responsible for the clearing of its synchronisation flag on all of the other processors. The remote processor can clear the flag after using the flag for synchronisation. With this method a two stage synchronisation is possible. However this solution is costly in the use of the computer backplane for communication. A remote processor must both busy wait on a number of remote flags and then clear them.

The use of busy waiting on remote processor flags is inadvisable on the Bath University Parallel Computer. The continual testing of the flag locations by remote processors can cause a processor 'deadlock'. The continual testing of a synchronisation flag by the remote processor prevents the local processor having access to its local bus and updating its flag. The busy waiting on the backplane also causes processor memory failure due to memory refresh failure. This is

described in detail in chapter 3. The semaphore 'deadlock' and the refresh failure can be prevented by putting idle states into all the processor busy wait loops. This allows the backplane to be released and used by other processors. It also allows each local processor to gain access to its bus. The addition of idle states however slows the speed of synchronisation and it relies on an empirical knowledge of the processor instruction execution times. The use of busy wait loops using the backplane will also give a general loss in computer backplane communication performance and slow all of the synchronisation routines.

6.5.3.4. The choice of synchronisation routines for the block predictor corrector integration algorithm.

In the design of the processor quartet synchronisation routines the speed of synchronisation was considered to be of greatest importance. To indicate the full possibilities of the block predictor corrector parallel integration method the synchronisation and communication time between processors should be kept to a minimum. The routines for processor synchronisation can be specific and do not need to be general purpose in nature. The method of synchronisation based on waiting on local flags was therefore chosen. Synchronisation using semaphores is more elegant and general purpose. However the number of processor actions in semaphore synchronisation is considerably higher. The greater number of actions also gives a heavier computer backplane utilisation. The signalling of a semaphore on a remote processor requires a backplane read and write as well as the testing of a remote flag to gain access to the semaphore. The setting of each flag in the synchronisation method based on writing on local flags requires just one backplane write. The synchronisation method based on waiting on remote flags has been discounted for the reasons given in section 6.5.3.3.

6.5.4. The practical implementation of the synchronisation routine based on waiting on local flags.

For the synchronisation routine based on waiting on local flags each synchronising processor in the processor quartet has two flag areas, each of one thirty-two bit word length. Each byte in the word is a flag for each processor in the quartet. All processors in the quartet have access to one of these byte flags on every processor. The use of byte rather than bit flags ensures that the operations on the flags by different processors are determinant. The setting of bit flags would involve the reading and writing of the other flags. On synchronisation each processor sets its byte flag on all the other processors. Each processor then busy waits on its own flag area until all the flags in its area are set. The flag area is then cleared and the processor proceeds. It is important that the flag area is local to the busy waiting processor as a system where processors check flags on all processors in the quartet causes processor 'deadlock'. Two synchronisation routines are used with each being used in turn. Both of the synchronisation routines uses a different flag area with each synchronisation routine protecting the flag area of the other from being cleared prematurely.

6.5.5. The practical implementation of the block predictor corrector steplength adjustment algorithm on the parallel computer.

To maximise the parallelism in the block predictor corrector method each processor in the quartet calculates its error and steplength adjustment factor. A common data area, which is placed on the fourth processor of the processor quartet, is used to determine the maximum error and the steplength adjustment factor for the next block calculation. Each processor in the block compares its

error value with the maximum error and updates this and the steplength adjustment value if its error is found to be greater. The processors then synchronise and read the maximum error and the steplength adjustment value for the next block calculation. Each processor determines from the block maximum error whether the last block was successful.

The maximum error location has to be initialised to zero for each use of the block predictor corrector integration algorithm. This initialisation is difficult. The maximum error location cannot be cleared until all the processors have read it. This cannot be guaranteed until the processor synchronisation after the reading of the maximum error location. However the clearing of the maximum error location after the next synchronisation could affect the calculation of the maximum error for the next block. Therefore the method of processor synchronisation used after the calculation of maximum error has been slightly modified. This modification allows the correct or determinant clearing of the block maximum error location. The synchronisation routine on the fourth processor of the block, which is used after the reading of the maximum block error, has therefore been changed. Normally it would first set the synchronisation flags on all the processors in the block. Instead it waits for its local synchronisation flags to be set. Thus it synchronises with all the other processors without allowing the processors to proceed. This ensures that the maximum error location has been read by all the processors in the block calculation. The fourth processor can then clear the maximum error location before setting its synchronisation flags on all the other processors. This allows the block calculation to proceed and the simulation to continue.

6.5.6. The results from the block predictor corrector method.

The four point block predictor corrector method was tested with the simple analytically solvable problems given in equations 6.10a and 6.10b. The results were compared with those for a modified Euler predictor corrector formulae solution of the same equations. Figures 6.5a and 6.5b compare graphically the simulation solutions for the block predictor corrector method, the Euler predictor corrector method and the correct analytical solution. In figure 6.5a the block predictor corrector method is compared to the Euler predictor corrector method for an initial steplength of 0.1 for the equation $dy/dx = cos(x)$ with $y(0) = 1$. Both methods take approximately the same integration time. However the Euler method shows a significant error for the calculation compared to the block predictor corrector method and the analytical solution. Figure 6.5b shows the same calculation for an initial steplength of 0.01. The Euler method is now closer to the analytical solution. However it is now ten times as slow as the block predictor corrector method for the same initial integration steplength.

6.6. Changes to the Diesel engine equations needed when using a parallel integration method.

With the successful use of the block predictor corrector method for the simple equations described in section 6.5.6, the method was used for the solution of the Diesel engine model equations. The Diesel engine model used was of a single cylinder engine with constant manifold conditions. This allowed the effect of the parallel integration method to be analysed in isolation without the inter-volume communication of a multi-volume model. The single cylinder model was also the only model that was implementable on the six processor computer that was then being used for the research for this thesis.

The Diesel engine equations used for both the serial and the parallel intergration methods are essentially the same. However certain changes need to be made in the method of calculation to take account of the parallel nature of the integration solution. These are described in the next two sections.

6.6.1. The calculation of ignition delay for a multi-processor integration algorithm.

The calculation of ignition delay in a cylinder control volume is based on the calculation of the average temperature and pressure in the cylinder after the injection of fuel. The equation for ignition delay is given in equation 6.11.

$$id = \frac{3.52 \times 10^{-3}}{P_m^{1.022}} e^{\frac{2100}{T_m}} \qquad \text{Equation 6.11.}$$

In the block predictor corrector method the values used to calculate the pressure and temperature averages are distributed across four processors. It is possible to allocate one of these processors to the calculation of ignition delay and for all of the other processors in the block calculation to read this result. However, to ensure maximum parallel efficiency, all processors have been made to contribute to the calculation of ignition delay. The average values of pressure and temperature are derived from the integral sums of the pressure and temperature values since fuel injection. Each processor calculates its effect on the temperature and pressure integral since fuel injection. All processors then sum the contributions from every processor in the block calculation and the previously calculated integral. From this they calculate the pressure and temperature averages. Maximum parallel computational throughput is therefore achieved.

Two calculation conditions are possible in the calculation of the pressure and temperature integrals:

- The fuel has been injected during the present block calculation.
- The fuel has been injected during a previous block calculation.

In the former the contribution of each processor to the average value depends on the proximity of the point to the injection point. The contributions of each of the block points to the calculated integral in both cases is shown in figure 6.6.

The next section gives all of the equations for the calculation of the ignition delay pressure and temperature integrals for all the block points. For each equation a reference is made to the diagramatic representation of the integral contribution in figure 6.6.

When the fuel injection angle has occurred during the present block the following contributions (ic) are made to the pressure and temperature (y) integrals:

- Any block point whose angle is less than the injection angle and is more than one steplength away from the injection angle makes no integral contribution (CASE A).

- For the block point immediately prior to the fuel injection angle the integral contribution is given by equation 6.12a (CASE B).

$$\begin{aligned} \theta_s &< \theta_i,\ \theta_i - \theta_s < h \\ ic &= \frac{y_s}{2}(\theta_s + h - \theta_i)\left(1 - \frac{(\theta_i - \theta_s)}{h}\right) \end{aligned} \qquad \text{Equation 6.12a}$$

- For the block point immediately after the fuel injection angle the integral contribution is one of two cases. If the block point is the last point of the block then the contribution is given by equation 6.12b (CASE C) otherwise it is given by equation 6.12c (CASE D).

$$\begin{aligned} \theta_i &< \theta_s,\ \theta_s - \theta_i < h,\ s = 4 \\ ic &= \frac{y_s}{2}(\theta_s - \theta_i)\left(2 + \frac{(\theta_i - \theta_s)}{h}\right) \end{aligned} \qquad \text{Equation 6.12b}$$

$$\begin{aligned} \theta_i &< \theta_s,\ \theta_s - \theta_i < h,\ s = 1,2,3 \\ ic &= \frac{y_s}{2}\left(h + (\theta_s - \theta_i)\left(2 + \frac{(\theta_i - \theta_s)}{h}\right)\right) \end{aligned} \qquad \text{Equation 6.12c}$$

- All the remaining internal block points have an integral contribution given by equation 6.12d (CASE E).

$$\begin{aligned} \theta_i &< \theta_s,\ \theta_s - \theta_i > h,\ s = 1,2,3 \\ ic &= y_s \cdot h \end{aligned} \qquad \text{Equation 6.12d}$$

- If the block point is the last point of the block then the integral contribution is given by equation 6.12e (CASE F).

$$\begin{aligned} \theta_i &< \theta_s,\ \theta_s - \theta_i > h,\ s = 4 \\ ic &= \frac{y_s}{2} \cdot h \end{aligned} \qquad \text{Equation 6.12e}$$

When the block being calculated is after the fuel injection point the following contributions are made to the ignition delay temperature and pressure integrals:

- For all the points in the block, except the last, the contribution is given by equation 6.12f (CASE G).

$$\begin{aligned} s &= 1,2,3 \\ ic &= y_s \cdot h \end{aligned} \qquad \text{Equation 6.12f}$$

- For the last point in the block the contribution from both the last point in the present block and from the last point in the previous block must be made. This gives a contribution given by equation 6.12g (CASE H).

$$\begin{aligned} s &= 4 \\ ic &= \frac{y_s + y_{s-4}}{2} \cdot h \end{aligned} \qquad \text{Equation 6.12g}$$

The average values for both pressure and temperature during ignition delay are given by equation 6.12h.

$$\overline{y} = \frac{\sum_0^s ic}{\theta_4 - \theta_i} \qquad \text{Equation 6.12h}$$

6.6.2. Combustion start for a multi-processor integration algorithm.

The simulation has to be started from the combustion angle after the calculation of ignition delay. All the processors know the crankshaft angle for combustion start as they all have calculated the ignition delay. The cylinder fuel-air ratio and cylinder mass are also known as these are constant during the closed half of the engine cycle. However the cylinder control volume temperature is unknown at

the combustion point. It would be possible to calculate the cylinder volume temperature on all the processors. However this would lead to an unecessarily large use of the computer backplane. It is therefore better to allocate this calculation to one processor. The temperature result can then be read by the other processors.

The processor with the crankshaft angle less than and closest to the combustion angle is used to calculate the volume temperature by extrapolation. The formula for this extrapolation is given in equation 6.13.

$$T_c = T_s + \left(\frac{\theta_c - \theta_s}{h}\right)(T_{s+1} - T_s) \qquad \text{Equation 6.13}$$

The calculated temperature value at combustion is written by the calculating processor into the temperature result area of the fourth processor of the quartet. The temperature result location of the fourth processor is read by all the processors in the processor quartet on the next calculation step. The choice of processor for the extrapolation calculation is made to reduce computer backplane usage. The processor used has access to most of the variables used in the extrapolation calculation in its local memory.

6.7. The block predictor corrector method applied to the Diesel engine equations.

Following the successful implementation of the block predictor corrector method for the simple problems, the method was applied to the Diesel engine equations. Initially the equations for just the cylinder induction period of the single cylinder model were tested. The results from this were compared to those for the modified Euler predictor corrector method. A number of problems were

experienced in the use of the block predictor corrector method for the cylinder induction equations and these are outlined in the following sections.

6.7.1. The problem of steplength adjustment.

When the block predictor corrector method was used for the Diesel engine equations the steplength adjustment scheme suggested by Birta and Abou-Raita [50] was found to give problems. At certain stages of the solution it produced a continual looping of the integration formulae. When these continual loops ocurred the maximum error of the block was found to be just greater than one. The step adjustment factor was therefore just less than one or at one. Numerical inaccuracy caused it to be calculated as one. Thus the equation steplength was not changed and a perpetual calculation loop was caused.

Since this research was undertaken a subsequent paper by Birta and Abou-Raita [55] has been published. In this they discuss the problem of continual looping of the integration formula for certain differential equations when the calculated block error is close to one. They suggest the addition of a constant multiplying factor into the steplength adjustment scheme. This ensures that the effects of numerical accuracy do not cause this continual looping. Their paper suggests a multiplying factor of 0.9

The use of lookup tables in the single cylinder simulation will also effect the steplength adjustment scheme. This will be particularly noticable in the calculation of mass flows through the cylinder valves. Because of the use of lookup tables, the valve mass flows are constant between points in the lookup table. Integration methods which adjust steplength by small amounts to gain stability will not therefore have any effect as a change will only be achieved

when the lookup table point used in the calculation is changed. It should be noted that for the Euler predictor corrector method, used in this research, adjustment of the integration steplength is always by halving. Therefore the integration step can always be made to use a lookup table point.

6.7.2. The problem in the calculation of the error value for the block predictor corrector method.

Each control volume in the Diesel engine model has three differential equations for the variables temperature, fuel-air ratio and mass. In the parallel single cylinder simulation each of these variables is solved using the block predictor corrector method. The magnitude of the three control volume variables varies considerably. There is therefore a problem of ensuring equal weight is given to the error calculation from each of the equation variables when determining the stability of the cylinder volume solution. The weighting of each variable error calculation is determined by a user defined value of error margin. Incorrect choice of this value leads to instability in one of the equations being ignored when apparent stability is indicated by another. In such a case undue prominence is being given to the error calculation of one of the cylinder variables.

Consideration has been given to normalising the error for each control volume variable using the maximum value of that variable. However this is felt to be ineffective as the maximum values for all the cylinder volume variables occur at different points in an engine cycle. The use of the maximum value may therefore be inappropriate to the size of the variable during certain stages of the engine cycle. A more effective method may be to normalise the simulation results with the maximum values of each variable in each engine calculation sector. In the Euler predictor corrector method the stability is determined by the ratio of the

last two integration formula iterations rather than the difference between them. This eliminates the problem of normalisation. The use of a ratio error calculation would therefore eliminate the problem of unequal error weighting. However it would also mean the steplength adjustment scheme suggested by Birta and Abou-Raita [50] could not be used.

6.7.3. The use of the one step block integration method with a fixed steplength.

Using the block predictor equations for solving the Diesel engine model equations it was found that to ensure continual solution of the equations the initial integration steplength used had to be made very small. This was both to avoid continual looping and numerical instability in the block predictor corrector method. This instability at large step lengths in parallel methods is noted by Kerkchoffs [47]. He notes that parallel integration algorithms may only be appropriate to certain problems. The use of a small integration steplength for the block predictor corrector formulae gave an inferior performance to the Euler predictor corrector scheme for the same single cylinder Diesel engine model.

When using the block predictor corrector integration method there is a problem of equation startup. The integration method requires several calculated solution points before it can be used. Therefore a one step block method was used to obtain these starting values. An analysis of the integration results for the startup using the one step method indicated that the results for the one step method were more stable and could use a larger steplength than the block predictor corrector method.

Therefore the parallel one cylinder Diesel engine simulation was changed to use the one step block integration method. Using this the stability of the Diesel engine model was found to be greater and a larger integration steplength could be used. This confirmed the results indicated by the initial analysis of the startup equations. The block one step method was implemented using a fixed integration steplength. The length of this fixed integration steplength can easily be altered through the engine cycle to take account of the change in the Diesel engine equation complexity. The block one step method can be applied iteratively to improve the integration solution. However it was found that the application of the one step block method equations only once gave comparable stability to the Euler predictor corrector method.

The use of the one step block method also allowed the processor synchronisation routines to be simplified. The clearance of the block maximum error was no longer necessary. The synchronisation method used by the fourth processor could be made the same as the other synchronisation routines.

6.7.4. Practical results for the single cylinder one step block implicit model.

The block one step method was successfully implemented for the whole of the engine cylinder cycle. The simulation results for an engine cycle for the block one step method is given in figure 6.7.

Using the block one step method a speed up of about two has been achieved. However further work will need to be carried out into the effect of integration steplength on the solution. The method has only been used at one engine condition point. The effect of different engine conditions on integration stability

and steplength will need to be fully investigated.

6.8. A discussion of how the block predictor corrector method can be applied to á full six cylinder Diesel engine simulation.

The one step block predictor corrector method has been shown to give a speed up of about two using four processors. With the present MC68020 parallel computer it is not feasible to use such a method for a larger Diesel engine simulation as the method is only of use when there are enough processors to give one processor per block point calculation. It is envisaged that a multi-control volume model may be developed with four processors for each engine control volume. The larger engine simulation requires not only communication between processors in each block but also between each block. Points in each block would however only need to communicate with the equivalent points in the other control volume blocks. The complete system would therefore proceed as two distinct inter-processor communication systems:

- Communication between each integration block member,
- Communication between the engine control volumes for each block point.

The communication in the latter is similar to that for the parallel Diesel engine simulation described in chapter 4.

6.9. Conclusions.

The increased parallelism identified in this chapter can only partially be implemented on the present MC68020 based parallel computer. This is because of the limitations in the number of processing elements and the inter-processor communication speed. The new Transputer parallel computer, at present being developed for Diesel engine simulation, will have a much larger number of processing elements and a higher inter-processor speed. The present Diesel engine simulation parallelism based on engine control volumes will only use a fraction of the number of processing elements in the new Transputer computer. Only increased parallelism in the Diesel engine simulation will allow the full power of the new computer to be obtained.

Figure 6.1 A diagram illustrating the possibilities for calculation of the cylinder control volume sub-equations in parallel.

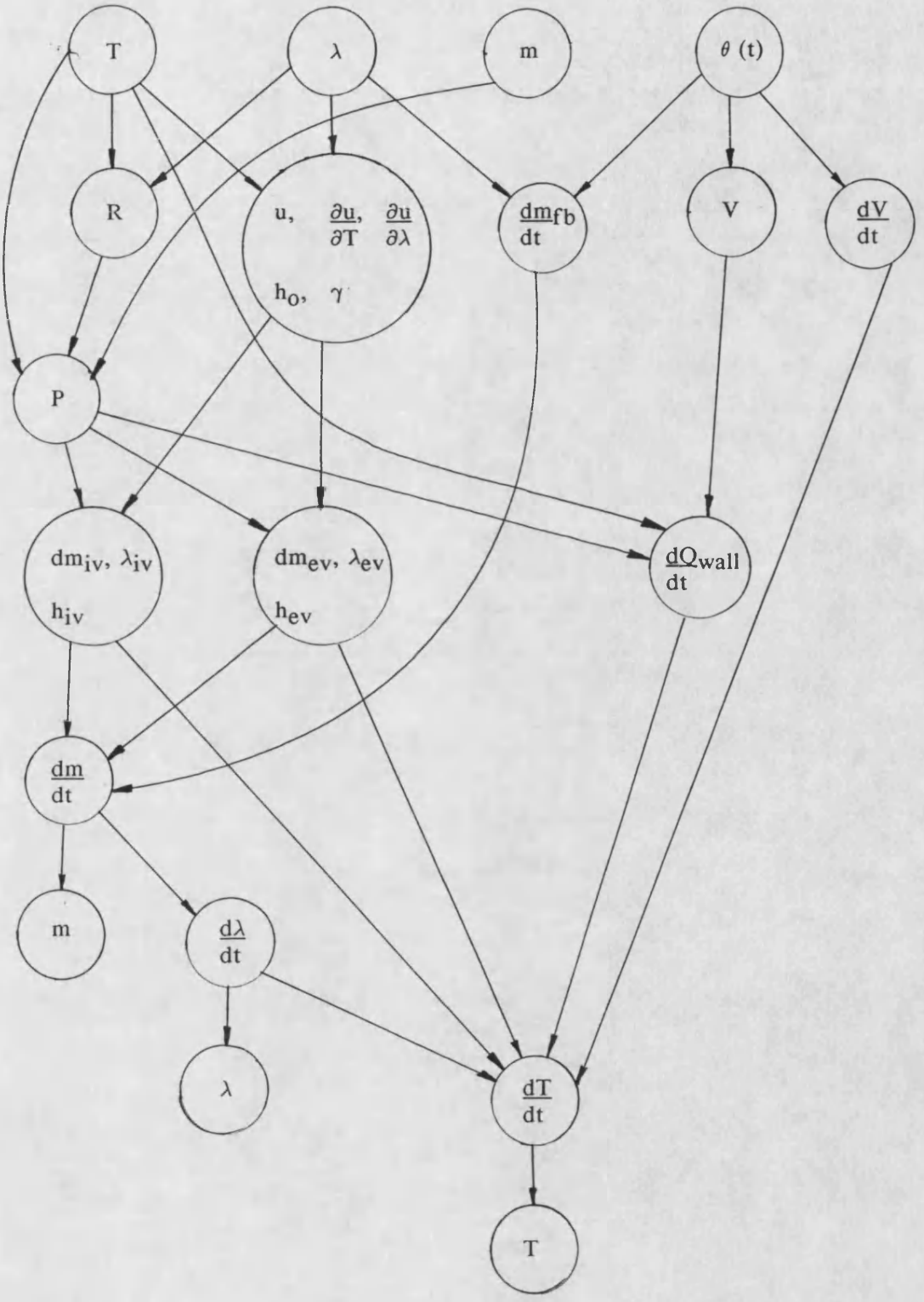


Figure 6.2 A flow diagram for the heat transfer calculation and the gas properties calculation in parallel.

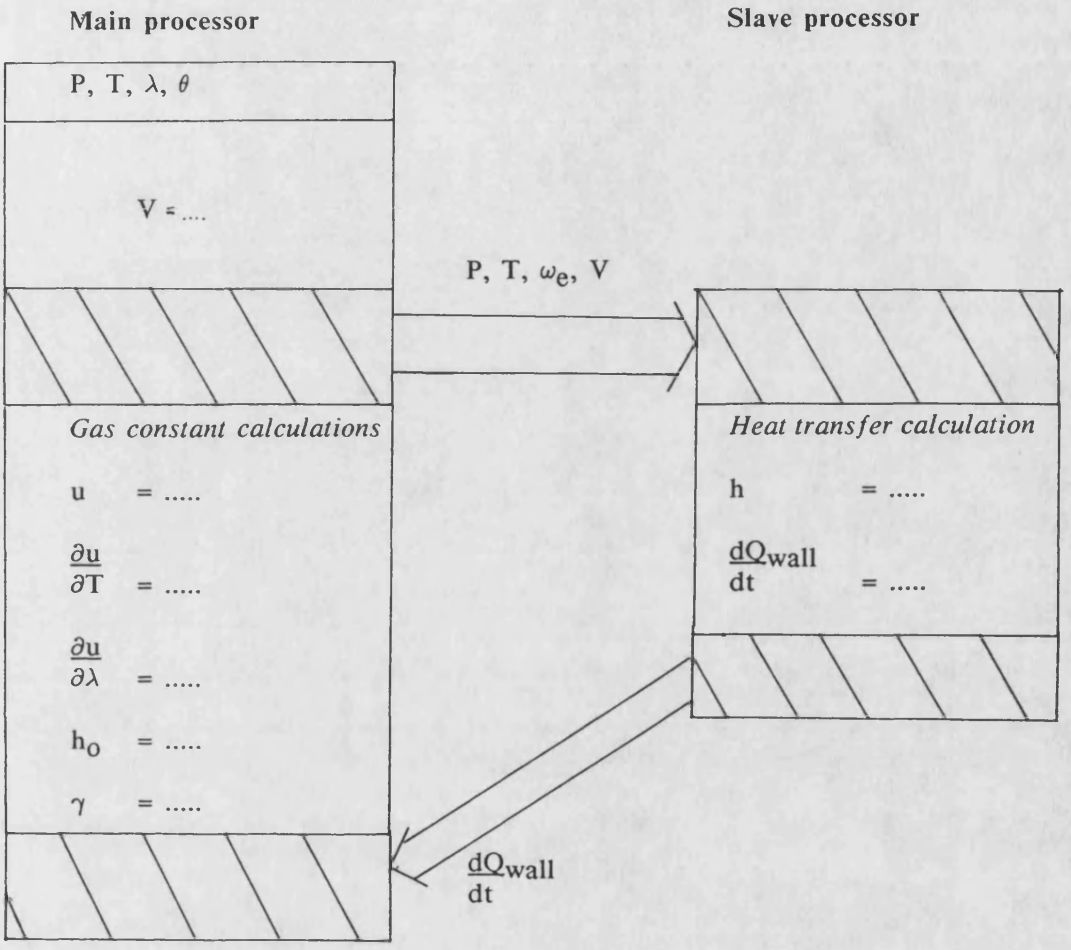


Figure 6.3 Data flow diagrams for the Miranker and Liniger method equation 6.1 (Figure 6.3a) and the Euler predictor corrector method equation 6.2 (Figure 6.3b). The diagrams indicate the widening of the computation front.

Figure 6.3a

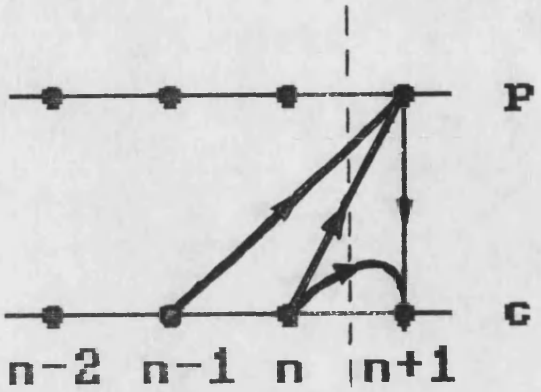


Figure 6.3b

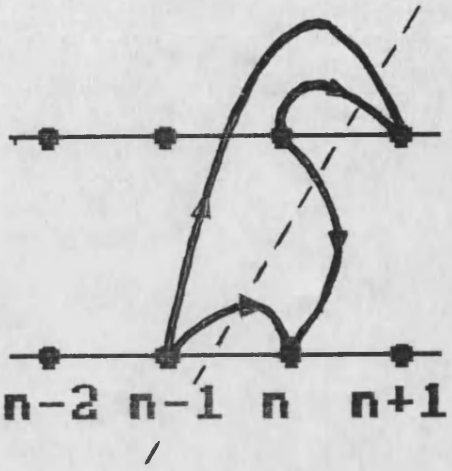


Figure 6.4 The sequence of computation for the Miranker and Liniger integration method (equation 6.1.)

$$
\begin{aligned}
y^p_{n+1} &\longrightarrow f^p_{n+1} \longrightarrow \\
y^c_n &\longrightarrow f^c_n \longrightarrow \\
y^p_{n+1} &\longrightarrow f^p_{n+1} \longrightarrow y^c_{n+1} \longrightarrow f^c_{n+1} \longrightarrow
\end{aligned}
$$

Figure 6.5a A comparison of the integration results for the block predictor corrector method and the modified Euler predictor corrector method for the equation $dy/dx = cos(x)$, $y(0) = 1$. Figure 6.5a shows the results for an initial steplength of 0.1.

```
d y d x  =  c o s ( x )  ;  y ( 0 )  =  1 . 0
```

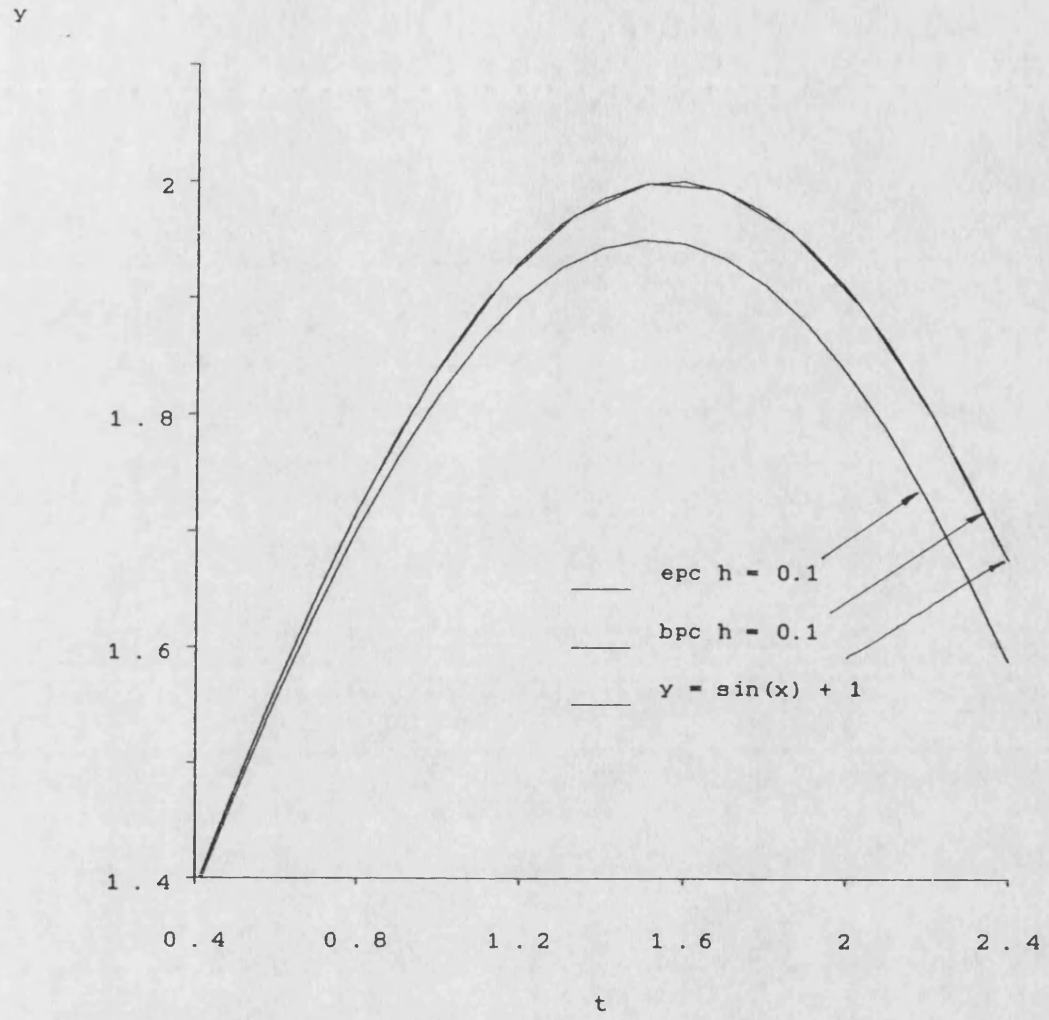


Figure 6.5b A comparison of the integration results for the block predictor corrector method and the modified Euler predictor corrector method for the equation $dy/dx = cos(x)$, $y(0) = 1$. Figure 6.5b shows the results for an initial steplength of 0.01.

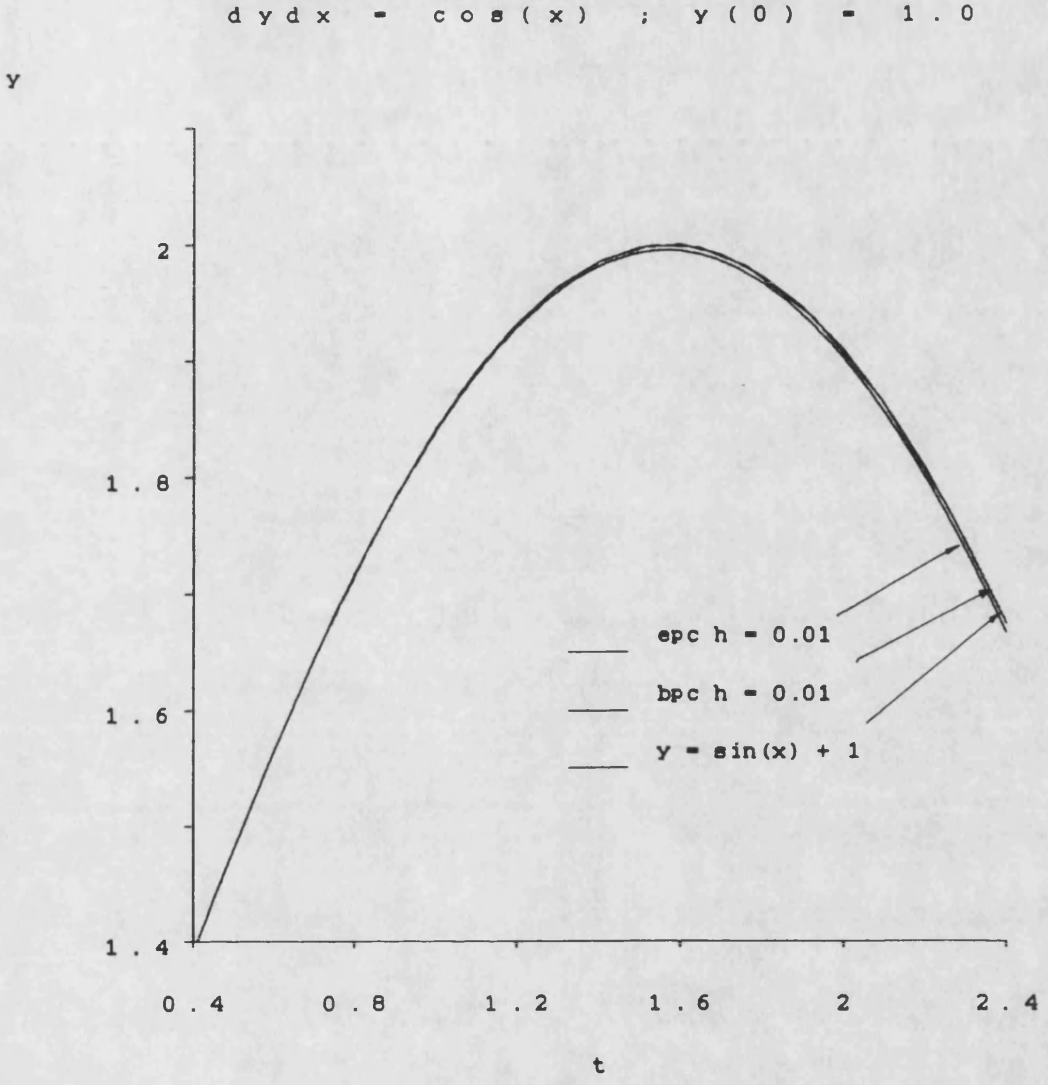


Figure 6.6. The integration contributions from each calculated block point for the calculation of the average temperature and pressure values for the calculation of ignition delay.

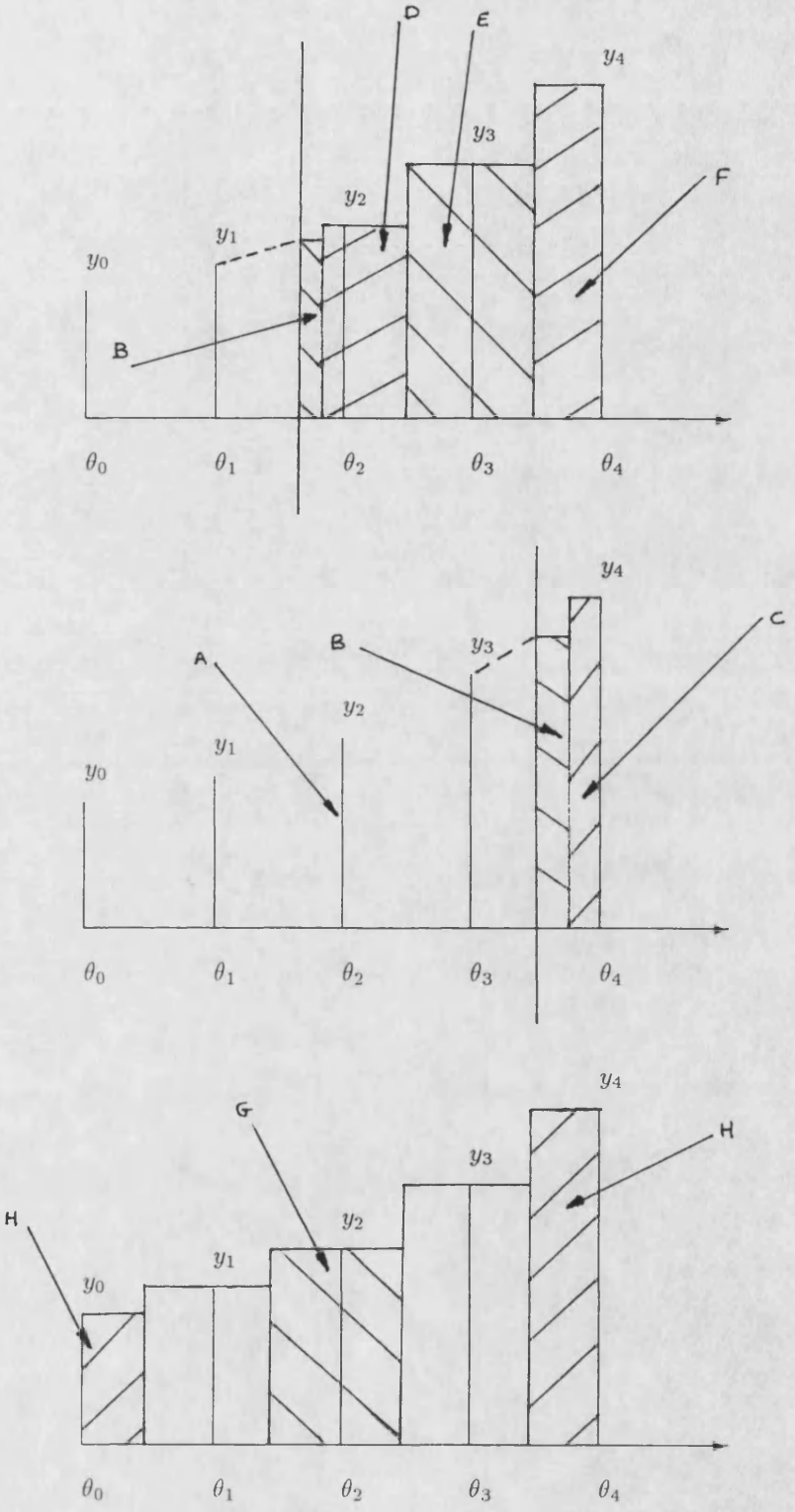


Figure 6.7a The results for the cylinder pressure for a complete cylinder simulation cycle for the block one step method.

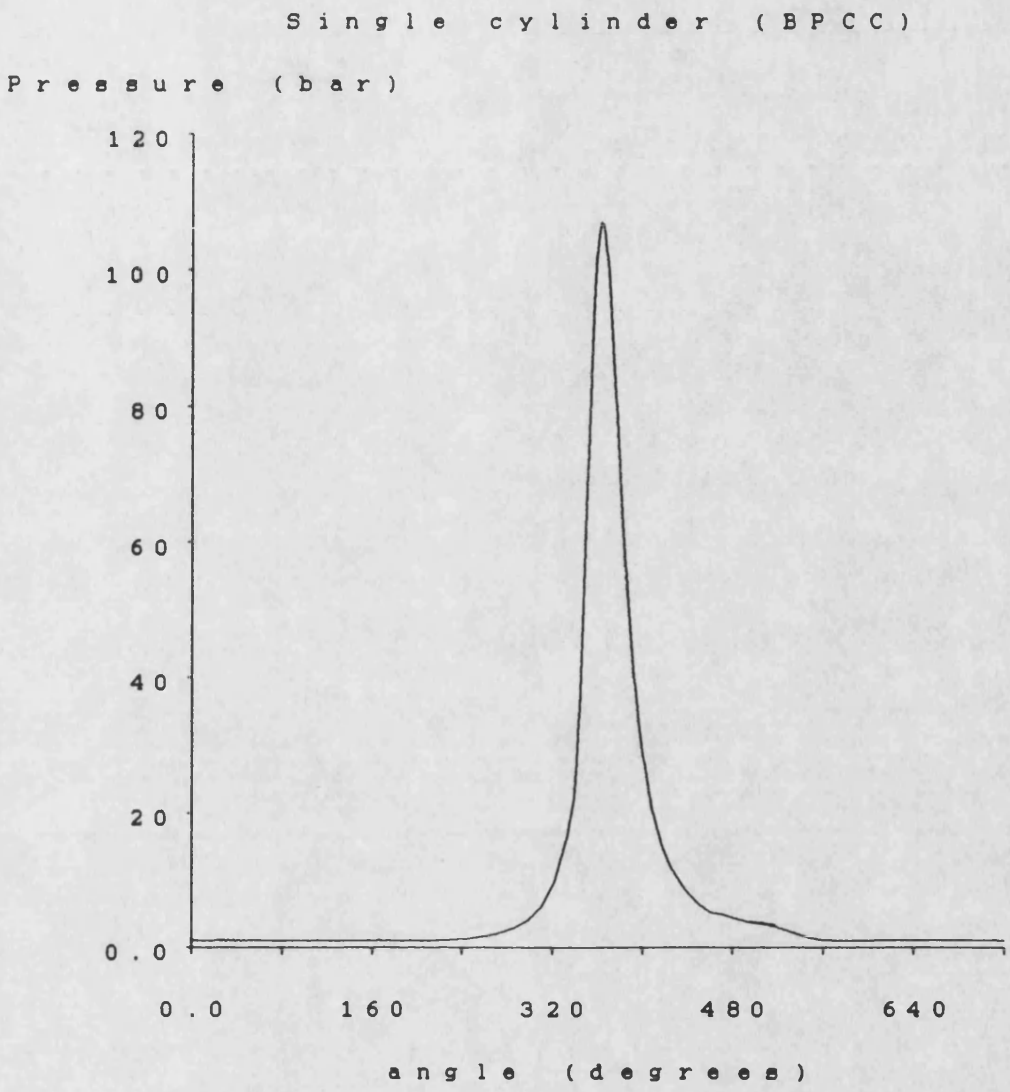


Figure 6.7b The results for the cylinder temperature for a complete cylinder simulation cycle for the block one step method.

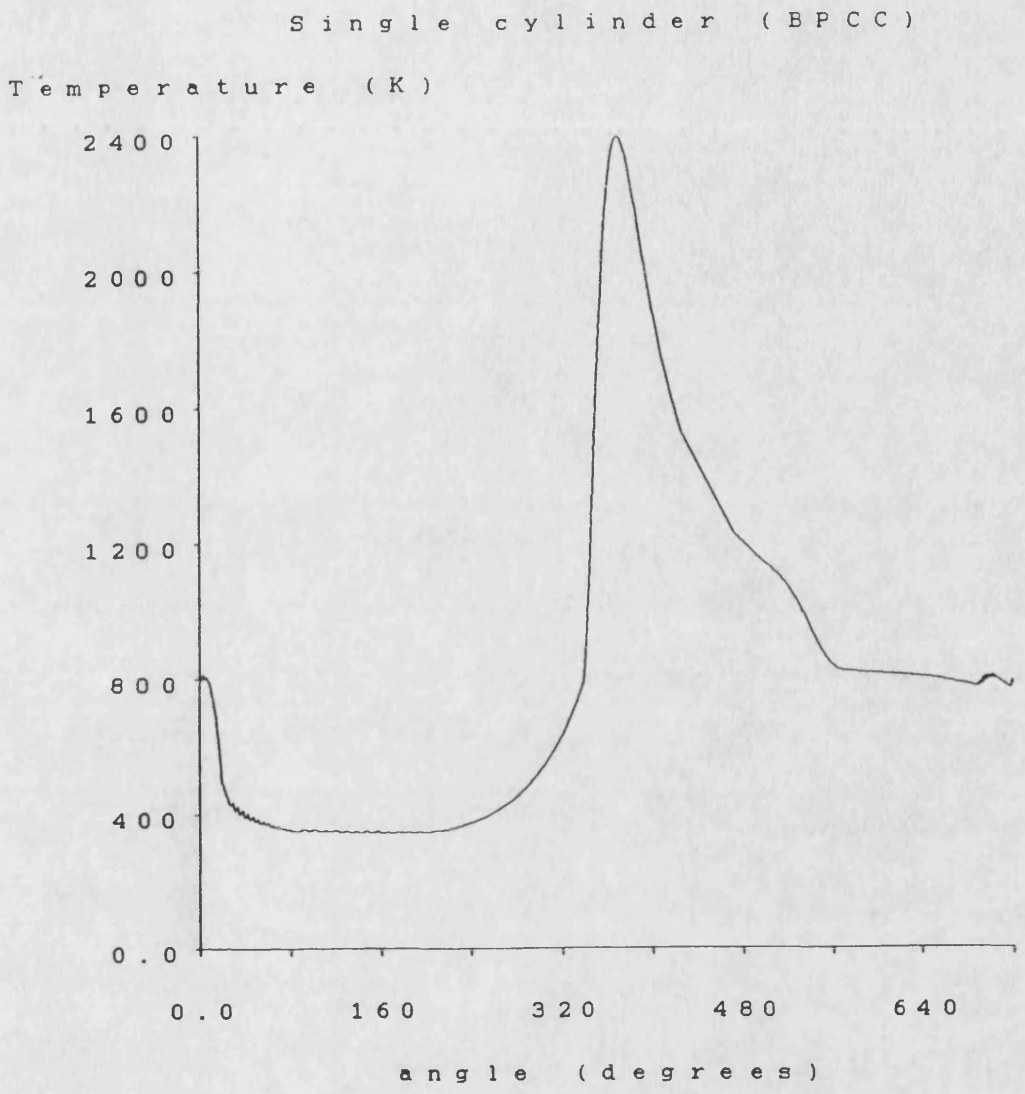


Figure 6.7c The results for the cylinder fuel-air ratio for a complete cylinder simulation cycle for the block one step method.

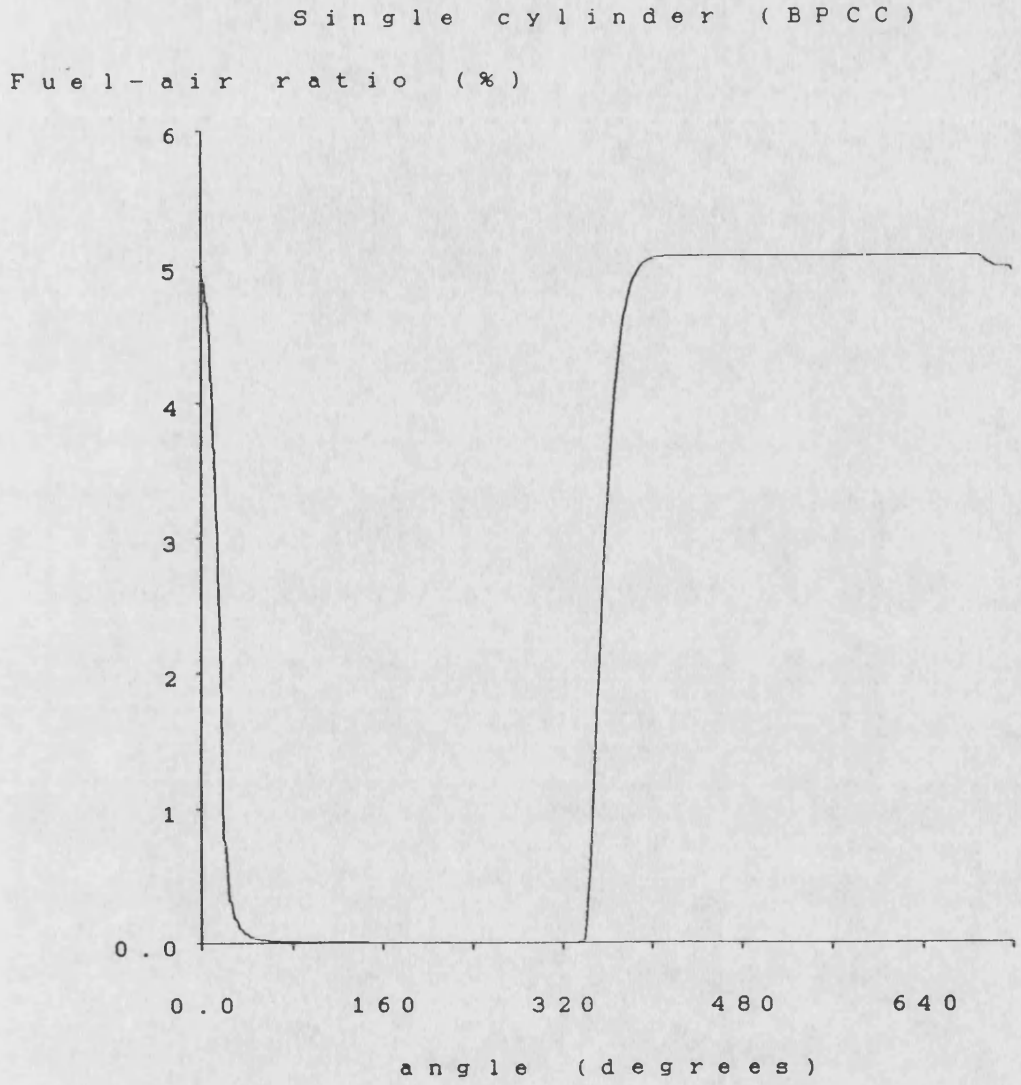
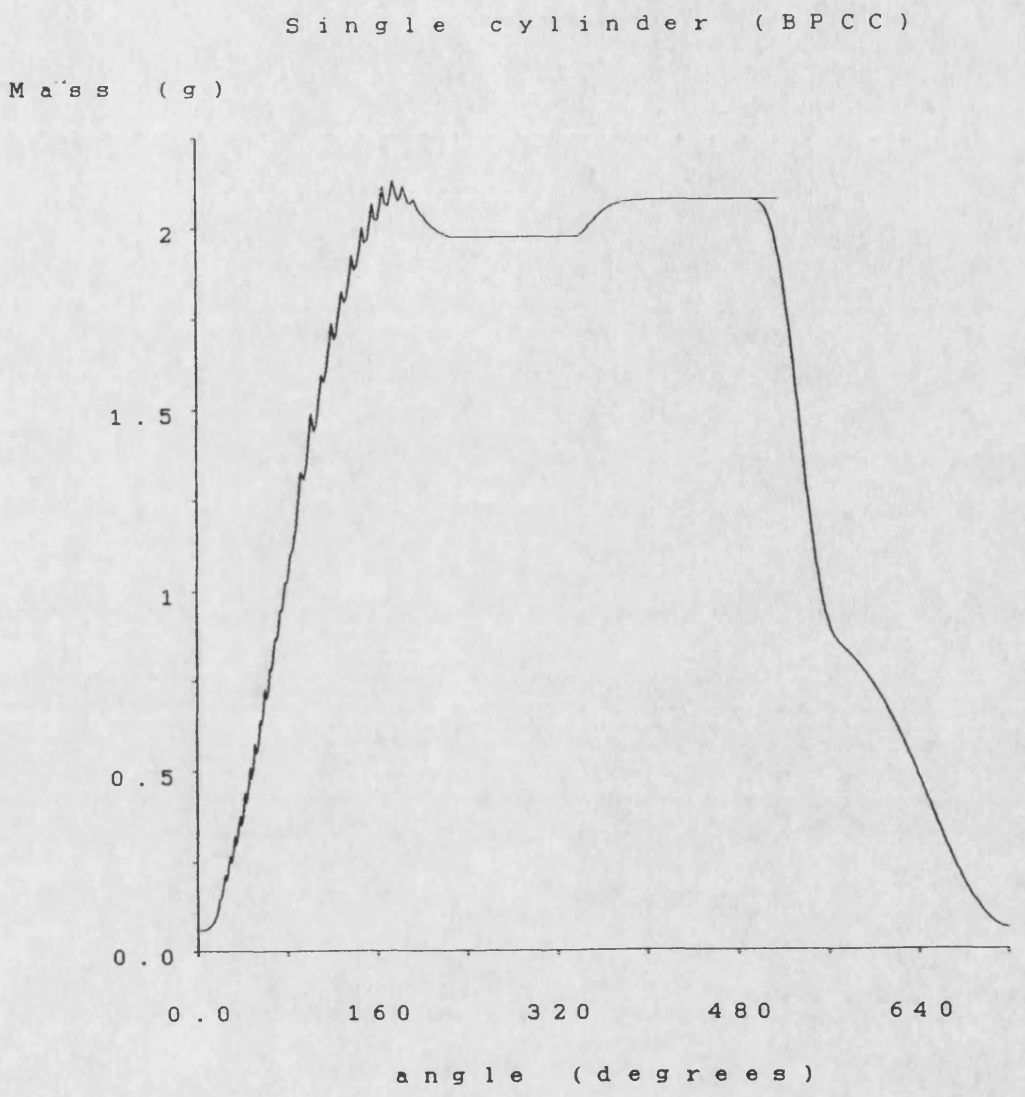


Figure 6.7d The results for the cylinder mass for a complete cylinder simulation cycle for the block one step method.



CHAPTER 7.

THE INSTRUMENTATION AND DATA ACQUISITION SYSTEM FOR THE TL11 TRUCK DIESEL ENGINE.

7.1. Introduction.

In chapter 2, a parallel Diesel engine simulation based on the Leyland TL11 truck engine was described. This chapter describes the Leyland TL11 engine in the Wolfson laboratory of the School of Mechanical Engineering at the University of Bath. This engine has been thoroughly instrumented for the purpose of validating the parallel Diesel engine simulation in this thesis. Subsequent to this the instrumented engine will be used in the investigation of engine fault conditions for use in condition monitoring. A discussion of engine condition monitoring is given in chapter 9. The present chapter describes the instrumentation used on the TL11 engine. It also describes the computer data acquisition system which was designed as part of the research for this thesis.

7.2. The Leyland TL11 truck engine.

The Leyland TL11 is a high speed Diesel engine used in trucks and buses. Figure 7.1 shows the TL11 engine in the Wolfson laboratory. The engine has six 'in-line' cylinders and has a volumetric capacity of 11.1 litres. It is a four stroke engine with a maximum power output of 190kW at a maximum speed of 2100rpm.

The TL11 engine has direct fuel injection into the engine combustion chambers. The fuel injection is driven by a fuel pump which is shown in figure 7.2. The fuel pump, driven at half engine speed from the engine crankshaft, is controlled in two ways: in the amount of fuel injected into the engine, the rack control, and in the time at which the fuel is injected into the engine, the timing control. The fuel pump is controlled hydraulically.

The Wolfson laboratory TL11 engine has been fitted with a variable geometry turbocharger which is shown in figure 7.3. The turbocharger has a normal compressor section which supplies air to the TL11 engine inlet manifold through an intercooler which is shown in figure 7.4. The charge air is water cooled in the intercooler. The turbine section of the turbocharger has a variable geometry. The variable geometry allows some control of the turbine characteristics and hence the supply of air to the engine. It is believed that the variable geometry is achieved through the use of a variable area nozzle ring immediately upstream of the turbine. However the precise manner in which it is achieved is confidential to the company who supported the development of the turbine. The variable geometry ring is controlled by a hydraulic actuator which can adjust the reduction in the turbine nozzle area from 0% to 50%. The turbine is supplied with exhaust gases from the two exhaust manifolds of the engine through a single entry into the turbine.

The power from the TL11 engine in the Wolfson laboratory is absorbed by a hydrostatic dynamometer. The dynamometer is shown on the right of figure 7.5. The dynamometer can absorb power from the engine in three modes: constant speed, constant torque and with torque proportional to the square of the engine speed which is known as 'windage'.

A complete description of the Leyland TL11 engine and its behaviour is given by Roberts [18]. A brief description of the main parts of the TL11 engine system is given in appendix C.

7.3. The instrumentation of the Leyland TL11 engine.

The TL11 engine has been extensively instrumented to give as full a picture of the operation of the engine as possible. All of the signals from the engine are fed through trunking out of the engine test cell to an instrumentation rack. This rack contains all the transducer signal conditioning electronics, the engine control electronics and the engine safety monitoring electronics. The instrumentation rack is shown in figure 7.6. A full description of all the instrumentation on the engine and the interface electronics is given in the Leyland TL11 instrumentation manuals [56].

The following sections describe the transducers that have been attached to the TL11 Diesel engine, with a brief description of the purpose of each of the transducers.

7.3.1. The cylinder pressure transducers.

The dynamic measurement of the engine cylinder pressure is important for the analysis of the process of combustion in the engine. The TL11 engine has been equipped with two Kistler 6121 piezo-electric pressure transducers and their associated charge amplifiers. A brief specification of these is given in table 7.1. The pressure transducers have been mounted at the top of cylinders three and six. Due to the extreme conditions in the cylinders, the pressure transducers cannot be used continuously in the cylinders. They are therefore removable to allow the deposits from fuel combustion to be removed. The pressure transducers have a fast response and can therefore measure the changes in cylinder pressure through the engine cycle. The two cylinder pressure transducers are marked as numbers 5 and 6 in the pressure transducer schematic diagram, figure 7.7.

7.3.2. The injector needle lift transducers.

The needle lift transducers indicate the dynamic timing of the injection of fuel into the cylinders of the engine. The needle is the non-return-valve in the cylinder fuel injector which prevents the gas from the cylinder passing back into the fuel pump. When fuel is injected into the cylinder, the needle is forced open by the pressure from the fuel pump. The needle lift transducers monitor the movement of the needle. Needle lift transducers have been fitted to cylinders three and six. The needle lift transducer is an inductive-capacitive displacement transducer. The transducer is driven locally by an oscillator box to reduce problems with long cables. The oscillator box drives an FM amplifier which is in the instrumentation rack outside the test cell.

7.3.3. The A.V.L. optical engine crankshaft angle transducer.

To monitor the crankshaft position of the engine and to derive the engine speed, an A.V.L crankshaft optical encoder type 360C/600 has been mounted on the front of the TL11 crankshaft. The encoder has two rings of dark and bright bands which are monitored by two sets of infra red transmitters and receivers. The first ring has 600 bright-dark bands and this gives a crankshaft angle resolution of 0.6^{o}. The second ring has just one bright mark which has been adjusted to give an output when the engine cylinder one is at top dead centre. The signal from the first ring is frequency multiplied using a phase locked loop to give an output pulse for every 0.2^{o} of engine revolution.

7.3.4. The top dead centre (TDC), the firing top dead centre (FTDC) and camshaft transducers.

On the flywheel side of the TL11 engine a number of magnetic pickups have been mounted. One of these is used to detect the engine top dead centre (TDC) position. The engine flywheel has a metal tag fixed to it which when it passes the TDC magnetic pickup indicates that cylinder one is passing through top dead centre. The remaining cylinders are at multiples of 120^{o} relative to this cylinder. The engine flywheel also has 360 teeth. Five magnetic pickups monitor this toothed wheel. At present only one of these is connected and this is used to derive the engine speed for the engine instrumentation rack safety electronics. The remaining pickups have been fitted to allow improved accuracy detection of the passing of each flywheel tooth. These will be used to cross reference the engine speed measurement derived from the AVL optical engine crankshaft transducer described in section 3.3.

The TDC signal is conditioned by the instrumentation rack electronics to give a TTL compatible pulse output. The TL11 engine is a four stroke engine and does two engine revolutions per engine cycle. It will therefore pass through top dead centre twice per cycle. To determine which half of the cycle the engine is in, a digital TTL level signal is generated from the engine camshaft. The camshaft rotates once for each engine cycle. The signal from the engine camshaft transducer is logically gated with the TDC signal to obtain a firing top dead signal (FTDC) which indicates the top dead centre at which combustion takes place in the first cylinder of the engine.

7.3.5. Turbocharger speed transducer.

The turbocharger rotational speed is measured optically. The turbocharger rotor has a small hole drilled through it. On either side of this hole a light emitting diode and a photo detector have been mounted. When the hole passes the LED, light falls on the detector and generates a signal. This signal is buffered to give a TTL compatible digital output. Two output pulses are generated for each turbine shaft revolution and from this the turbine rotational speed may be calculated.

7.3.6. The pressure transducers.

Pressures in the engine are measured at the inlets and outlets of the major components of the engine. A schematic representation of the placing of the pressure transducers on the engine is given in figure 7.7. The numbers in brackets in the following lists of pressure transducers refer to the numbers in figure 7.7. Two types of pressure transducer have been fitted to the TL11 engine. The Furness differential pressure transducer, type FCO40 Mk.2 is used for small pressure differences. A brief description of this transducer is given in table 7.2. The following pressures are measured using the differential pressure transducers:

- The differential pressure across the air flow orifice. (1)
- The compressor inlet pressure relative to atmospheric pressure. (2)
- The turbine outlet pressure relative to atmospheric pressure. (8)

All of the air introduced into the TL11 engine is passed through an orifice within the engine air box. By measuring the pressure differential across the orifice the rate of air flow into the engine can be calculated.

For the higher absolute pressures in the engine, the Druck PDCR 811 general purpose pressure transducer has been used. The details of this transducer are given in table 7.3. The following pressures are measured using these transducers:

- The compressor outlet pressure. (3)
- The inlet manifold pressure. (4)
- The rear exhaust manifold pressure. (7)

All of the steady state engine pressure transducers are mounted outside the engine test cell and are connected to the measuring points using plastic tubing. Due to the length of this tubing, the pressures measured by the transducers can only represent pressure averages. The pressures from the engine are also monitored by manometers with which the results from the pressure transducers may be compared.

7.3.7. The engine temperature transducers.

The temperatures in the engine are measured using Cromel-Alumel (K type) thermocouples. These are robust and are used all over the engine. The thermocouples are buffered using conditioning cards. Two types of conditioning cards have been used and these are described in table 7.4.

The conditioning cards have sixteen thermocouple inputs with iso-thermal termination. These are multiplexed onto an amplifier with auto cold-junction compensation using a platinum resistor. The voltage outputs from the conditioning cards are fully linearised. The output of the card is selected using a four bit binary address and a 1ms latching pulse. The cards have been interfaced to allow the user to present temperature readings to a display on the

engine instrumentation rack or to remotely read the channels using a computer.

A schematic representation of the placing of the major temperature transducers on the TL11 engine is given in figure 7.8. The numbers in brackets in the following lists of temperature thermocouples refer to the numbers in figure 7.8. The following temperatures are monitored by the two thermocouple conditioning cards:

A. The 0C to 1000C Signatrol TCK5021a thermocouple card.

1. The front exhaust manifold temperature. (12)
2. The rear exhaust manifold temperature. (11)
3. The compressor outlet temperature. (3)
4. The turbine outlet temperature. (13)
5. The exhaust stack temperature. (14)
6. The Celesco smoke meter temperature.
7. The dynamometer hydraulic oil cooler water inlet temperature.
8. The dynamometer hydraulic oil cooler water outlet temperature.
9. The hydraulic actuator servo temperature.

B. The -50C to 100C Signatrol TCK5021b thermocouple card.

1. The engine coolant inlet temperature. (9)
2. The engine coolant outlet temperature. (10)
3. The air box orifice inlet temperature. (1)
4. The air box orifice outlet temperature. (2)
5. The engine test cell temperature.
6. The engine fuel temperature after the fuel lift pump.
7. The engine oil sump temperature.
8. The inlet manifold temperature 1. (7)

9. The inlet manifold temperature 2. (8)
10. The engine oil cooler water outlet temperature.
11. The intercooler air outlet temperature. (6)
12. The outside temperature.
13. The fuel tank temperature.
14. The intercooler water inlet temperature. (4)
15. The intercooler water outlet temperature. (5)

7.3.8. The Exac fuel flow meter.

The Exac fuel measurement system gives a continuous reading of fuel flow into the engine. The system consists of a EXAC EX12 mass flow sensor and the EX 8100 flow transmitter. The transmitter is an intelligent processing unit which for this application has been configured to give a display read out of fuel flow and a current output proportional to fuel flow. This signal is conditioned to give a ±10V output signal. The details for the system are given in table 7.5.

7.3.9. Engine torque meter.

The torque generated by the TL11 engine is measured by a Vibro-meter TG-220/B in-line torque meter which is mounted at the rear of the engine between the engine and the dynamometer. The torque transducer uses rotary transformers. The details of the torque transducer are given in table 7.6.

Table 7.1 The Kistler 6121 piezo-electric pressure transducer and Kistler 5007 charge amplifier.

Linearity	±0.05%
Accuracy over range.	±1%
Frequency range	0-180kHz

Table 7.2 The Furness low range differential pressure transducer, Type FCO40 Mk.2.

Input range	0 - 25mbar
Accuracy	± 1%
Response time	50 ms
Thermal stability	±0.5% per 10C

Table 7.3 The Druck general purpose pressure transducer, PDCR 811.

Input range	0 - 3.5 bar
Linearity	±0.1%
Thermal stability	±1.5% in -20C - 80C

Table 7.4. The Signatrol thermocouple conditioning cards.

The Signatrol TCK 5021a conditioning card.

Temperature range	0 - 1000C
Accuracy	± 1C
Output	0 - 10V

The Signatrol TCK 5021b conditioning card.

Temperature range	0 - 100C
Accuracy	± 0.5C
Output	0 - 10V

Table 7.5 The EXAC EX12 flow sensor and 8100 flow transmitter.

Mass flow range	0.907 - 136.05 gs^{-1} (0.12-18 lb/min)
Accuracy	0.15%
Response time	0.1 - 50 s (adjustable)

Table 7.6 The Vibro-meter TG-220/B torque meter.

Torque range	0 - 2000 Nm (4000 Nm maximum)
Linearity	± 0.1%
Frequency range	0 - 1000 Hz
Speed range	0 - 28,000 rpm

7.4. The engine data acquisition system hardware.

To monitor, record and present results from the TL11 engine a computer data acquisition system has been designed and built as part of the research in this thesis. The following sections describe the design aims for such a system and describes in detail the acquisition system designed.

7.4.1. The specification for the engine data acquisition computer.

The TL11 engine uses a large range of transducers with varying outputs. These outputs are pre-conditioned in the TL11 engine instrumentation rack and the outputs from the rack may be split into two groups:

1. Analogue voltage outputs in the range -10V to +10V

2. Digital TTL pulse train outputs, such as:
   a. Crankshaft angle.
   b. TDC, FTDC and camshaft.
   c. Turbine speed.

The analogue voltage outputs may be further sub-divided into three sub-groups:

1. Analogue outputs with a high frequency range which require a high acquisition rate, such as:
   a. The in-cylinder pressures.
   b. The needle lift outputs.

2. Analogue outputs with a lower frequency range which require a slower acquisition rate, such as:

   a. The fuel mass flow rate.

   b. The engine gauge pressures.

   c. The engine differential pressures.

   d. The engine hydraulic actuator control outputs.

3. Analogue outputs from the thermocouple conditioning cards which require external selection of their output.

The engine data acquisition computer has been designed to handle all of these types of output.

Using the high speed transducers, it is possible to analyse the process of combustion in the engine. For this analysis, a data acquisition rate of at least once per 0.5^{o} of crankshaft rotation is important [57]. The TL11 engine has a maximum speed of 2100 rpm and to give 0.5^{o} resolution at this speed a high speed acquisition rate of 25kHz is needed.

7.4.2. The computer system.

The data acquisition system has been based on the MC68000 single board computer system designed by Dale [23] which is described in chapter 3. The computer uses standard Bath University computing boards and two new data acquisition interfaces, for analogue and digital data from the engine. The engine data acquisition computer has been designed using the following standard boards:

1. A 'master' MC68000 processor board.

The 'master' processor board provides the following facilities:

a. The computer terminal interface.
b. The computer floppy disc interface.
c. The computer 80Mbyte hard disc interface.
d. The transfer of data from the analogue data interface to the memory.

2. A 'slave' MC68000 processor board.

The 'slave' processor board provides the following facilities:

a. The transfer of data from the digital data acquisition interface to the memory.
b. The memory for the storage of data from the digital data acquisition interface.
c. The 25kHz data acquisition clock for both the analogue and the digital data interface.

3. A two megabyte back-plane memory board.

The back-plane memory board is used for the following:

a. The multi-processor TRIPOS processor work queues.
b. The memory for the storage of data from the analogue data acquisition interface.

4. A Multilink [31] local area network board.

The Multilink board is used for the communication of results between the acquisition computer which is sited in the Wolfson laboratory and the simulation computer in the School of Electrical Engineering.

5. An EFCIS colour graphics board.

The EFCIS graphics board is used for the display of collected data by the acquisition computer on a colour graphics screen.

7.4.3. The analogue data acquisition interface.

In the design of the analogue data acquisition interface the following points were considered important:

1. The number of analogue input channels should be easily expandable to allow for a possible increase in the number of transducers on the engine.

2. The analogue interface should allow two data acquisition rates, a high speed acquisition rate of 25kHz and a lower acquisition rate for slower transducers.

3. Because the analogue signals from the engine are pre-conditioned to be in the range -10V to +10V, the analogue interface should be general purpose and useable in applications other than the TL11 engine.

4. Any increase in the number of analogue input channels should not cause a degradation in the high speed data acquisition rate of 25kHz.

5. The analogue to digital convertors used should be greater than 10 bits wide. This will give a small error due to digitising and will allow low level input signals to be interfaced more easily.

6. The analogue input circuitry to the analogue interface should reflect the accuracy of the analogue to digital convertors used.

An analogue interface has been designed based on standard analogue data acquisition boards. These standard boards have been designed to be used in multiples to obtain the required number of analogue inputs. Figure 7.9 shows one of the analogue data acquisition boards. Each analogue data acquisition board has three analogue to digital (A/D) convertors. These A/D convertors are the twelve bit Analog Devices AD574J [58] which have an eleven bit accuracy. Two of the A/D convertors on each board are used directly for fast data acquisition. The input to the third A/D convertor is multiplexed using the National Semiconductor LF13508 analogue switch to give eight slower analogue inputs. The fast data acquisition channels have a conversion rate of 25 kHz and the slow data acquisition channels have an effective data acquisition rate of 3.125 kHz.

7.4.3.1. The choice of sample and hold amplifier for the analogue boards.

The AD574 is a successive approximation type A/D convertor with its own internal clock. In the analogue acquisition boards, the A/D convertor is used in its free-running mode and is started by a pulse on its R/C input. The subsequent

conversion time of the AD574 is variable with a maximum conversion time of 35μs. The completion of the conversion is indicated by the AD574 status output. The input to each of the A/D convertors uses a National Semiconductor LF358 [59] sample and hold amplifier. This holds the analogue voltage input to the A/D convertor steady during the A/D conversion. The use of a 25kHz acquisition rate gives a 40μs conversion time. With a maximum A/D conversion time of 35μs this allows a minimum sample time for the sample and hold amplifier of 5μs. The hold capacitor of the sample and hold amplifier has therefore been chosen to give a sample time of 5μs with 0.001% accuracy. The 0.001% accuracy is compatible with the eleven bit accuracy of the A/D convertor.

The timing diagram for a AD574 conversion in free running mode is given in figure 7.10. This figure also shows the associated timing for the LF358 sample and hold amplifier in the analogue data acquisition boards.

7.4.3.2. The analogue board input anti-aliasing filters.

To prevent aliasing of the analogue signals, the inputs to the two fast channels and the eight slow input channels are all low pass filtered using a six-pole Butterworth filter. A six-pole Butterworth filter gives an aliasing error of 0.04% at the filter cut off frequency which is appropriate to the eleven bit accuracy of the A/D convertor.

7.4.3.3. The choice of data transfer method between the analogue acquisition boards and memory.

In a simple computer data acquisition system it is usual for data to be transferred from the A/D convertor to the memory using the microprocessor. The analogue acquisition board requests that its data be read by interrupting the processor. The rate of the data acquisition and the possibly large number of analogue channels precludes the use of the MC68000 with interrupt control for the transfer of data between the analogue data acquisition boards into memory. A minimum interrupt cycle time of 30μs is achieved under TRIPOS and one transfer of a sixteen bit value by the MC68000 takes 2μs. This would only allow a maximum of 5 transfers for each data acquisition cycle of 40μs. The analogue data acquisition boards have therefore been designed to use a direct memory access (DMA) controller for data transfer.

The MC68000 processor board has a DMA controller the HD68450 [26]. The HD68450 has four DMA channels, two of which are already used by the Winchester hard disc driver and the floppy disc driver on the 'master' processor board. The fourth DMA channel on the 'master' processor board has been dedicated to data transfer between the analogue acquisition boards and memory.

The HD68450 DMA controller allows two modes of data transfer between a device and memory and these are known as dual addressing and single addressing. In the dual addressing mode, both the data source and the data destination in the data transfer are MC68000 memory addresses. Data is read from one address into the controller and then written to the other address. Data transfer using single addressing is faster than using dual addressing as either the source or the destination device is directly controlled by the DMA controller. The device is

controlled using the DMA channel control lines of DMA request and DMA acknowledge. In a single addressing transfer, only one memory location is accessed and simultaneously data is transferred to or from the directly coupled device. The sequence of actions performed by all the components involved in a single addressing mode data transfer is given in figure 7.11.

The single addressing mode has been used for the data transfer between the analogue boards and the memory as it is the most efficient data transfer method and therefore allows the greatest possibility for expansion of the number of analogue input channels in the data acquisition system. The use of the fastest transfer method also means that the MC68000 on the 'master' processor board has a greater time to process data and display results when the DMA controller is not using the processor bus for data transfers.

7.4.3.4. The modification of the 'master' processor board to allow the use of the DMA single addressing mode on the back-plane.

To allow the use of the single addressing mode of the DMA controller with the analogue acquisition boards, a modification has been made to the 'master' processor board. When transferring data the DMA controller will access memory locations in the back-plane memory through the computer back-plane. This has the effect of enabling the back-plane data buffers of the 'master' processor board. The output of these buffers will therefore clash with the data buffers of the analogue acquisition board which will be enabled at the same time. The data on the back-plane will therefore be corrupted and this corrupted data will be placed in the back-plane memory. To avoid this, the back-plane buffer enable signal on the 'master' processor board has been changed to disable the backplane

buffer when a DMA acknowledge (DACK) is generated by the fourth channel of the DMA controller. This modification means that data from the analogue boards can only be read into back-plane memory and not directly into the memory of the 'master' processor. A circuit diagram of the modification to the 'master' processor board is given in appendix A.

7.4.3.5. The data transfer structure of the analogue data acquisition boards.

In the single addressing mode the DMA controller directly controls the data transfer device. All of the analogue input channels on all of the analogue acquisition boards have therefore been designed to appear as one directly readable device to the DMA controller on the 'master' processor. To achieve this, five new back-plane signals have been used which link all of the analogue acquisition boards and control the combined actions of them. These new signals are:

- STSO (Status output)
- STSI (Status input)
- PHBO (Previous board Has Backplane Output.)
- PHBI (Previous board Has Backplane Input.)
- ACLK (25kHz acquisition clock.)

Table 7.7 gives a list of the data acquisition computer backplane signals with the positions of the new signals shown in bold. The purpose of each of the signals is outlined in the following sections. Figure 7.12 shows the DMA timing diagram for a three analogue board system showing the sequence in which data is enabled onto the computer back-plane by the analogue boards.

The analogue acquisition boards are arranged in descending order of priority from left to right in the card frame of the acquisition computer. The STSO and the PHBO signals from each analogue board are wired respectively to the inputs STSI and PHBI of the next lower priority board. A schematic diagram of the back-plane wiring for the analogue acquisition system is shown in figure 7.13.

The STSO and STSI lines control the generation of DMA requests to the DMA controller by the analogue acquisition boards. A DMA request (DREQ) is only generated when all the analogue convertors in the system have completed their conversions. The STSI input to an analogue board when asserted indicates that all of the higher priority analogue boards have completed conversion. The STSO output from an analogue board indicates that this board and all of the higher priority boards have completed their conversions.

The PHBO and PHBI lines control the sequence in which the data from each of the analogue boards is transferred by the DMA controller into memory. Data is transferred from the analogue boards in order of board priority with the highest priority board having its three convertor outputs read first. The PHBI input of each board when asserted indicates that a higher priority board has still not been read. The PHBO output of each board is asserted when the DMA request is generated. The PHBO output is negated when all of the convertor channels on that board have been read.

This section outlines in detail the steps that occur on each data acquisition cycle for the analogue acquisition boards . The component numbers given refer to those on the analogue acquisition board circuit diagrams in appendix A. For each acquisition cycle the following sequence of actions occurs:

1. All of the A/D convertors are started by a pulse on their R/C inputs. This is derived from the system 25kHz acquisition clock (ACLK) using monostable C9.

2. The A/D convertors (D1,D2,D3) assert their status output (STS) which set the respective sample and hold amplifiers (F1,F2,F3) on their analogue input into hold mode.

3. When the conversion is complete, each A/D convertor negates its status output (STS) which puts the sample and hold amplifier into sample mode. The STS signal also latches the data output from each of the A/D convertors into the twelve bit latches (C2/C3, C4/C5, C6/C7).

4. The status outputs (STS) from all three convertors on each board are combined with the status signal from the higher priority boards (STSI) by the NAND gate B5a. This signal is outputed as the status signal (STSO), which is fed to the (STSI) input of the next lowest priority analogue board.

5. The assertion of the combined status output (STSO) is used to set the flip-flop A8b which generates a DMA request (DREQ). The DREQ signal output to the computer back-plane is through a removable link. The link is only connected on the least priority analogue board. The DREQ is therefore generated when all of the A/D convertors in the system have completed their conversions.

6. The assertion of the combined status output (STSO) sets the flip-flop A8a which sets the Previous Has Backplane Output (PHBO) of the board. This signal is connected to the Previous Has Backplane Input (PHBI) of the next

lowest priority board through the computer back-plane. The PHBI and PHBO signals are used to determine the sequence in which data is presented to the DMA controller by the analogue boards.

7. The fourth channel of the DMA controller on the 'master' processor board, has been programmed to operate in its burst transfer mode. The DMA channel will therefore (after arbitrating for the 'master' processor board bus) continue to generate DMA cycles as long as its DMA request (DREQ3) input line is asserted.

8. For each DMA cycle the DMA acknowledge signal (DACK3) is asserted. Each assertion of the DACK signal enables one of the A/D output latches on one of the analogue acquisition boards.

9. Data is transferred by the DMA controller from the highest priority analogue board first. The highest priority analogue board has its PHBI input permanently disabled by its input pull-up resistor. The PHBI signal is gated with the DACK signal on each analogue board. This prevents lower priority analogue boards from being read until the higher priority boards have been read.

10. On each analogue board, the PHBI/DACK signal is used to clock a shift register B6. This shift register enables each of the convertor latches (EN1,EN2,EN3) in turn onto the internal analogue board data bus (IDATABUS). The combined PHBI/DACK signal also enables the internal data bus output buffers (A4,B4a), thus placing the data onto the back-plane where it is transferred into the back-plane memory.

11. The assertion of the EN3 signal on the analogue board is used to clear the DREQ flip-flop (A8b). Only the DREQ signal from the lowest priority board is connected to the DMA controller. Therefore the DREQ signal is only negated when the last A/D convertor output is about to be read.

12. The subsequent negation of the EN3 signal on the analogue board clears the PHBO flip-flop (A8a) and negates the PHBO output. This indicates that all of the data from the A/D convertors on that analogue board has been transferred and that the next analogue board may now be enabled for data transfer.

It should be noted that while the data is being transferred from the analogue boards to the memory by the DMA controller, the next A/D cycle on all of the analogue boards can continue. This is because the outputs from the A/D convertors are latched and it is these latched outputs which are transferred by the DMA controller.

The circuit diagrams for the analogue convertor board are given in appendix A. The method of data transfer from the analogue boards to the memory means that data from each channel is separated by data from other channels. The structure of the analogue data when transferred into memory is shown in figure 7.14.

7.4.3.6. The interface to the engine thermocouple conditioning cards.

The thermocouple conditioning cards have a sixteen way multiplexed output. Therefore the reading of the individual temperatures on the conditioning card involves the external selection of which temperature is placed on the output of the card. In the design of the analogue acquisition boards, this thermocouple

selection has not been incorporated as a standard feature of the boards. Instead, one of the standard analogue acquisition boards has been modified to interface with the thermocouple conditioning cards. The circuit diagram for this modification is given in appendix A.

The output from the thermocouple conditioning cards is selected by a four bit selection code and a 1ms latching pulse. This allows a maximum rate of switching between the thermocouple inputs of 500Hz. The selection code and the latching pulse are derived from the 25kHz clock using a counter circuit (A9,A2). The outputs from the thermocouple conditioning cards are connected to the first two slow analogue data inputs on the highest priority analogue acquisition card. The slow analogue outputs have a data acquisition rate of 3.125kHz whilst the thermocouple inputs are switched at a rate of 390.625Hz. Therefore the outputs from the thermocouple cards are eight times oversampled.

The inputs to the slow analogue data channels are low pass filtered to prevent aliasing. However, these filters will have the effect of smoothing the outputs from the multiplexed thermocouple conditioning cards. Therefore, the anti-aliasing filters on the slow data channels, which are used for the thermocouple inputs, have been removed. The thermocouple inputs are directly connected to the eight way multiplexing switch (E2) on the analogue board.

7.4.4. The digital pulse train data acquisition board.

Unlike the analogue inputs, the digital inputs are very specific in nature. For this reason, the digital board is not general purpose and is only designed to handle the digital inputs from the TL11 engine. The digital board has been designed to read three signals from the TL11 engine. These are the output of the

A.V.L. angle encoder, the firing top dead centre (F.T.D.C.) pulse train, and the turbocharger shaft pulse train. From these two outputs are derived, an engine crankshaft angle counter and a turbocharger half revolution counter. The circuit diagrams for the digital acquisition board are given in appendix A. A picture of the digital acquisition board is shown in figure 7.15.

The digital acquisition board is used in conjunction with the 'slave' processor board of the acquisition computer. The 'slave' processor board has no connection to hard discs and so does not use its SASI interface. The MC68230 parallel interface timer (PIT) which is usually used for the SASI interface may therefore be used as a parallel port. To use the PIT directly some modifications have to be made to the MC68000 board and these are described in appendix A. The PIT on the 'slave' processor board is configured as a sixteen bit input port and is connected to the digital acquisition board directly through a forty way ribbon cable.

7.4.4.1. The engine crankshaft angle counter.

The crankshaft angle counter is incremented by the A.V.L optical encoder output and reflects crankshaft angle in multiples of 0.2^{o} from 0^{o} to 720^{o}. The crankshaft angle counter is cleared by the FTDC signal on every engine cycle. As the crankshaft angle count for a full engine cycle only requires twelve bits, the upper four bits of the counter have been used as an engine cycle counter. These four bits are clocked by the FTDC input.

The PIT sixteen bit input port is used to read the crankshaft angle counter. The angular position of the engine needs to be read at the same rate as the high speed analogue acquisition channels. The crankshaft angle counter is therefore

read by the PIT at a 25kHz acquisition rate. When the PIT is programmed in sixteen bit input mode, the reading of data is controlled by the assertion of the H3 input to the PIT. The H3 input is derived from the 25kHz data acquisition clock (ACLK). The PIT on the 'slave' processor board has been interfaced to generate DMA request cycles when data is read by the PIT. The PIT is memory mapped and so the DMA controller transfers the PIT value to memory using its dual addressing mode.

The incrementing of the crankshaft angle counter is asynchronous to the 25kHz data acquisition clock (ACLK). The digital board has therefore been designed to ensure that the crankshaft angle counter cannot be incremented at the same time as it is read by the PIT. This would cause inconsistent data because of the time delays in the counter circuit. When the incrementing of the crankshaft angle counter and the 25kHz negative clock edge coincide, the assertion of the PIT H3 line, and therefore the reading of the counter, is prevented until the crankshaft angle counter has been incremented. This is achieved using the shift register g5 which is clocked by the 8MHz system clock. The input to the shift register is the AVL pulse input. This pulse input is delayed by the shift register and the following functions are performed by the delayed signals:

1. The assertion of the AVL pulse input is used to disable the generation of an H3 PIT read output.

2. The AVL pulse input delayed by 250ns is used to clock the angle counter and possibly clear the crankshaft angle counter.

3. The AVL pulse input delayed by 1000ns re-enables the generation of an H3 PIT read output.

7.4.4.2. The turbocharger half revolution counter.

The turbocharger pulses are counted and are used in the derivation of turbocharger shaft speed. As the counter is not used to derive angular position, the turbocharger counter can be read less often than the crankshaft angle counter. The turbocharger counter is clocked by the turbocharger pulse input which occurs for every half revolution of the turbocharger. A 96.66 Hz clock, which is derived from the 25 kHz clock (ACLK), is used to latch the turbocharger counter for reading and to clear the turbocharger counter. The turbocharger counter is read by the 'slave' processor under interrupt control. From the 96.66 Hz turbocharger acquisition clock, a 'slave' processor interrupt cycle is generated using the PIT H4 handshake line. The PIT H4 input is programmed to generate a PIT interrupt to the 'slave' processor. The latched turbocharger counter is memory mapped onto the computer back-plane. When the 'slave' processor board is interrupted by the PIT the latched turbocharger counter is read and this result is written into the memory of the 'slave' processor board.

The turbocharger counter has been designed to ensure that the clocking of the turbocharger counter does not occur at the same time as the latching of the counter by the 96.66 kHz clock. This is achieved using a synchronous state machine (C3,C4). The state diagram for the turbocharger counter state machine is given in figure 7.16.

The circuit diagrams for the digital acquisition board are given in appendix A. A list of the PIT interface lines with the descriptions of how these are used for the interface to the digital acquisition board interface is given in table 7.8.

7.4.4.3. The system 25kHz data acquisition clock (ACLK).

Both the analogue and the digital acquisition boards use the 25kHz clock (ACLK) as the basis of their data acquisition cycles. This clock is generated by the timer section of the PIT on the 'slave' processing board. The PIT timer is programmed to produce a 25kHz square wave on the PIT TOUT output line. This signal is buffered on the digital acquisition board and this buffered signal drives the ACLK line on the computer back-plane. The presence of the clock on the backplane automatically starts acquisition cycles. Therefore the signal is controlled using the PIT clock input line TIN which is programmed as a clock enable signal. Data acquisition may be started by an internal software trigger or by an external trigger. The PIT PC0 output signal is used to enable the external trigger input. When enabled, the external trigger is latched by flip-flop G4b. The PIT PC1 output signal is used as an internal trigger input. The external trigger and the internal trigger (PC1) signals are combined and are used as the clear input to the flip-flop G4a. The clock enable output TIN is generated by the output of flip-flop G4a and is clocked by the FTDC pulse. The flip-flop is held cleared by the combined internal trigger and external trigger inputs. When either trigger signal is asserted, the flip-flop clear is negated and the acquisition clock is started on the next FTDC pulse. This synchronisation of the start of data acquisition with the FTDC pulse means that the data read by the acquisition system always starts at the same crankshaft angle.

It is often useful to read data from the engine acquisition system when the engine is not running. As the trigger input is started by the engine FTDC pulse, the triggering arrangement described above cannot be used when the engine is not running. However, the clock output may be programmed to start without reference to the TIN input. This triggering facility is programmed as part of the

digital device driver described in section 5.3. It is particularly useful when calibrating the engine instrumentation.

5. The engine data acquisition system software.

The data acquisition computer uses the operating system multi-processor TRIPOS [23]. The interface between the data acquisition boards and the operating system consists of two MC68000 assembly coded device drivers for the analogue and digital acquisition cards and a BCPL coded remote-device handler routine. The remote-device handler routine is for the control of the digital acquisition driver on the 'slave' processor board by the 'master' processor board. These interface routines are outlined in the following sections:

5.1. The use of array chaining to allow the DMA controller to handle large memory transfer counts.

The acquisition computer has a large back-plane memory for the storage of data from the ahalogue acquisition cards. This memory allows the storage of up to 1048576 16 bit data values. This data has to be transferred by the DMA controller which only has a sixteen bit memory transfer count (MTC) which only allows 65536 memory transfers. To allow larger transfers, DMA chaining has to be used, where the whole data transfer is made up of several smaller transfers. In both the analogue and the digital acquisition drivers, DMA array chaining is used. This is where the sequence of memory transfers is specified in a memory array. Each element in the array consists of a four byte memory address and a two byte memory transfer count. The structure of the chain array is shown in figure 7.17a. The DMA controller is programmed into array chaining mode. Its base address register points to the beginning of the memory array and its base

transfer count (BTC) is programmed with the number of elements in the chain array. When the DMA is started, the first memory address and memory transfer count in the chain array are read into the memory address register (MAR) and the memory transfer count (MTC) of the DMA controller. At the same time the BTC is decremented. When the MTC is exhausted, the next element from the chain array is read into the DMA controller. This continues until the BTC is exhausted.

For the analogue and digital drivers, each driver has a predefined chain array which is large enough for the transfer of up to 2097152 values. The chain array is pre-programmed for a series of maximum memory transfer counts of 65536. The device driver is responsible for the filling in of the memory address values and the last memory transfer count in the array. The last memory transfer count in the array represents the remainder of the memory transfer count when it has been divided into a number of 65536 transfer counts. An example of an array chain memory structure is shown in figure 7.17b where the values in the array which are filled in by the driver are shown in bold.

7.5.2. The analogue data acquisition board driver.

The driver initialisation. (INIT)

The following sequence of action is performed when the analogue acquisition driver is initialised:

1. The DMA controller channel four is set up for the following:
   (i) Burst transfer mode.
   (ii) Single addressing mode.

(iii) Sixteen bit transfers.

(iv) Transfers between device and memory with the memory counter being incremented.

2. The device driver start, stop and interrupt addresses are patched in the device driver DCB.

The driver packet handling routine. (START)

The TRIPOS device drivers are designed to handle communication packets from TRIPOS tasks. The analogue device driver is designed to handle only one type of packet, the acquisition packet. The structure of this packet is given in figure 7.18. The packet gives a BCPL pointer to the beginning of a data area in the back-plane memory and the length of this piece of memory and hence the number of data transfers that can occur. The back-plane memory is obtained using the global memory allocation routine getcomvec.

The device driver executes the following sequence of actions on receiving a packet.

1. The BCPL memory address is converted into a machine address.
2. The memory length in BCPL words is converted into the number of sixteen bit word transfers.
3. The number of word transfers is divided into the number of 65536 transfers and a remainder.
4. From the number of 65536 transfers the value of the DMA BAR array pointer is calculated.
5. The chain array memory address locations are filled using the machine

memory address as the starting address.

6. The DMA controller is set up for array chain type data transfer.

7. The analogue acquisition boards are reset.

8. Channel four of the DMA controller is started.

The driver interrupt handling routine.

The analogue device driver only handles one type of interrupt. This comes from the DMA controller and indicates that the data transfer has been completed. The interrupt routine does the following:

1. It clears the DMA interrupt.

2. It resets the analogue acquisition boards.

3. It reads the DMA controller error register and puts this in the result field of the acquisition packet.

4. It returns the analogue acquisition packet.

7.5.3. Digital data acquisition board driver.

The driver initialisation. (INIT)

The following sequence of action is preformed when the digital acquisition driver is initialised:

1. The DMA controller channel two is set up for the following:
   (i) Cycle steal transfer mode.
   (ii) Dual addressing mode.
   (iii) Sixteen bit transfers.

(iv) Transfers between a device and memory with the memory address counter being incrememted.

(v) The DMA device address register is set to point to the PIT port register.

2. The PIT device is set up for the following:

(i) As a 16 bit input port.

(ii) To generate DMA requests when the input port is read on the PIT H3 line being asserted.

(iii) To generate interrupt requests when the PIT H4 line is asserted.

(iv) To generate a 25khz square wave output from the PIT timer output.

(v) The internal trigger and external trigger enable outputs are disabled.

3. The device driver start, stop and the two device interrupt addresses for the DMA controller and the PIT device are patched into the driver DCB.

The driver packet handling routine. (START)

The digital device driver is designed to handle only one type of TRIPOS packet, the acquisition packet. The structure of this packet is given in figure 7.19. The packet gives two BCPL pointers to the beginning of two memory areas in the 'slave' processor board memory. It also gives the length of these areas of memory. The memory areas are for the storage of results from the angle counter and the turbocharger counter respectively. The memory is obtained using the memory allocation routine getvec. The last packet argument of the acquisition packet specifies the type of triggering that will be used to start the data

acquisition clock. The device driver executes the following sequence of actions on receiving a packet:

1. Both BCPL memory addresses are converted into a machine addresses.
2. The memory length in BCPL words is converted into the number of sixteen bit word transfers.
3. The number of word transfers is divided into the number of 65536 transfers and a remainder.
4. From the number of 65536 transfers the value of the DMA BAR chain array pointer is determined.
5. The chain array memory addresses are filled using angle counter machine address as the starting address.
6. The DMA controller is set up for array chain type data transfers.
7. The turbocharger data area pointer (TDAP) is initialised with the machine address of the turbocharger counter memory area.
8. The method of triggering the clock (ACLK) is programmed.
9. The DMA controller is started.

The driver interrupt handling routine.

The digital device driver handles two types of interrupt. These come from the DMA controller and the PIT driver. The interrupt from the PIT device indicates that the turbocharger counter should be read. An interrupt from the DMA controller indicates the completion of the angle counter data transfer. The DMA interrupt routine does the following:

1. The DMA interrupt is cleared.
2. The DMA error register is read and this is written in the result field of the acquisition packet.
3. The digital acquisition packet is returned.

The PIT interrupt routine does the following:

1. The PIT interrupt is cleared.
2. The value of the turbocharger data area pointer (TDAP) is read.
3. The turbocharger counter address is read and is stored in the address pointed to by TDAP.
4. The value of TDAP is incremented.

7.5.4. Digital data acquisition board remote device handler.

The digital board driver is loaded onto the 'slave' processor using the multi-processor TRIPOS routine REMDEV. The control of the data acquisition is all done by the 'master' processor board. However, the tasks on the 'master' processor cannot communicate directly with a remote device. They can only communicate with a task on the 'slave' processor. Therefore a digital acquisition board handler task has been written. This is loaded onto the 'slave' processor and is used to handle packets to the remote device from the 'master' processor. The remote handler is also designed to manage the memory on the 'slave' processor board which is used to store the results from the crankshaft angle and turbocharger speed counters.

7.5.5. Menu based data collection and processing software.

A menu based system has been developed for the Diesel engine acquisition computer using the data acquisition hardware and software described in this chapter [60]. This allows the user to easily record and display information from the engine. The system has the ability to monitor the running of the engine and to save complete logs of data. At present there are three analogue acquisition boards in the computer and therefore it is possible to store up to 4.66 seconds of information in one go using the two megabytes of back-plane memory. For an engine running at 1500 rpm this can allow the storage of up to 58 engine cycle.

Table 7.7 The Bath University Parallel Computer backplane pin descriptions.

	Edge connector J1		Edge connector J2	
Pin No.	Row a	Row c	Row a	Row c
32	+5V	+5V	+5V	+5V
31	-5V	-5V	+15V	-15V
30	D14	D15	FC0	DLY1
29	D12	D13	FC1	DLY2
28	D10	D11	FC2	DLY3
27	D8	D9	FC3	DLY4
26	D6	D7	/IAKOC	DLY5
25	D4	D5	/POR	DLY6
24	D2	D3		DLY7
23	D0	D1	/IRQ7	DLY8
22	/ASOLD		/IRQ6	/IOPG
21	EOLD	**ACLK**	/IRQ5	/68PG
20	/RESET	/BRO	/IRQ4	**STSO**
19	R/W	/DTACK	/IRQ3	/BR3
18	/VMA	/VPA	/IRQ2	/BR2
17	/BERR	/BPGA	/IRQ1	/BR1
16	/UDS	/LDS	/BGOUT0	/BGIN0
15	CLK	A20	/IAOUT	/IAIN
14	A18	A19	/BGOUT3	/BGIN3
13	A16	A17	/BGOUT2	/BGIN2
12	A14	A15	/BGOUT1	/BGIN1
11	A12	A13	/AS	ECLK
10	A10	A11	**/PHBO**	**/PHBI**
9	A8	A9	PCL3	**STSI**
8	A6	A7	PCL2	
7	A4	A5	PCL1	/HIBYTE
6	A2	A3	PCL0	/DONE
5	A21	A1	**/DREQ3**	**/DACK3**
4	A22	A23	/DREQ2	/DACK2
3	-12V	-12V	/DREQ1	/DACK1
2	+12V	+12V	/DREQ0	/DACK0
1	0V	0V	0V	0V

Table 7.8 The digital acquisition board PIT bus description.

Pin no.	PIT description.	Digital board description.
16	PB0	Angle counter D0
14	PB1	Angle counter D1
12	PB2	Angle counter D2
10	PB3	Angle counter D3
8	PB4	Angle counter D4
6	PB5	Angle counter D5
4	PB6	Angle counter D6
2	PB7	Angle counter D7
40	PA0	Angle counter D8
38	PA1	Angle counter D9
36	PA2	Angle counter D10
34	PA3	Angle counter D11
32	PA4	Cycle counter D0 (D12)
30	PA5	Cycle counter D1 (D13)
28	PA6	Cycle counter D2 (D14)
26	PA7	Cycle counter D3 (D15)
24	H1	Turbocharger count interrupt request.
22	H2	(unused)
20	H3	Angle count DMA request.
18	H4	Reset input.
9	PC0	Internal trigger.
11	PC1	External trigger enable.
13	TIN	System clock trigger.
15	TOUT	System 25kHz clock.

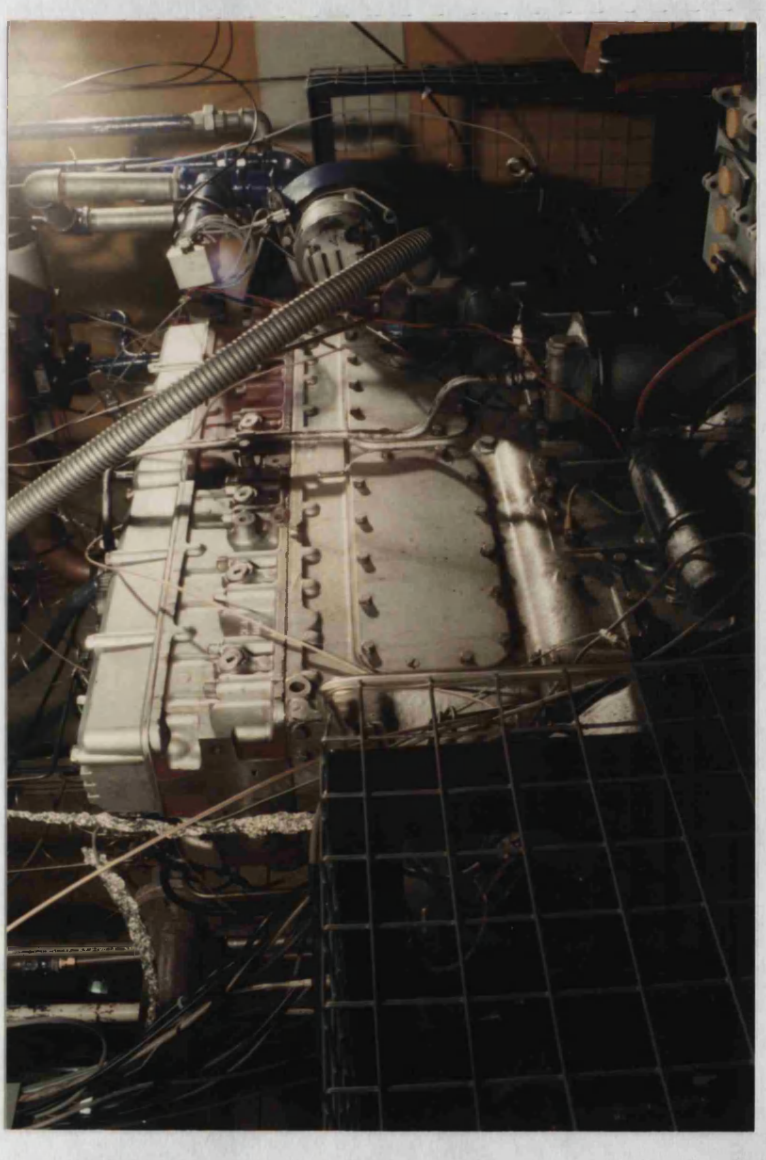


Figure 7.1. The Wolfson laboratory Leyland TL11 engine.

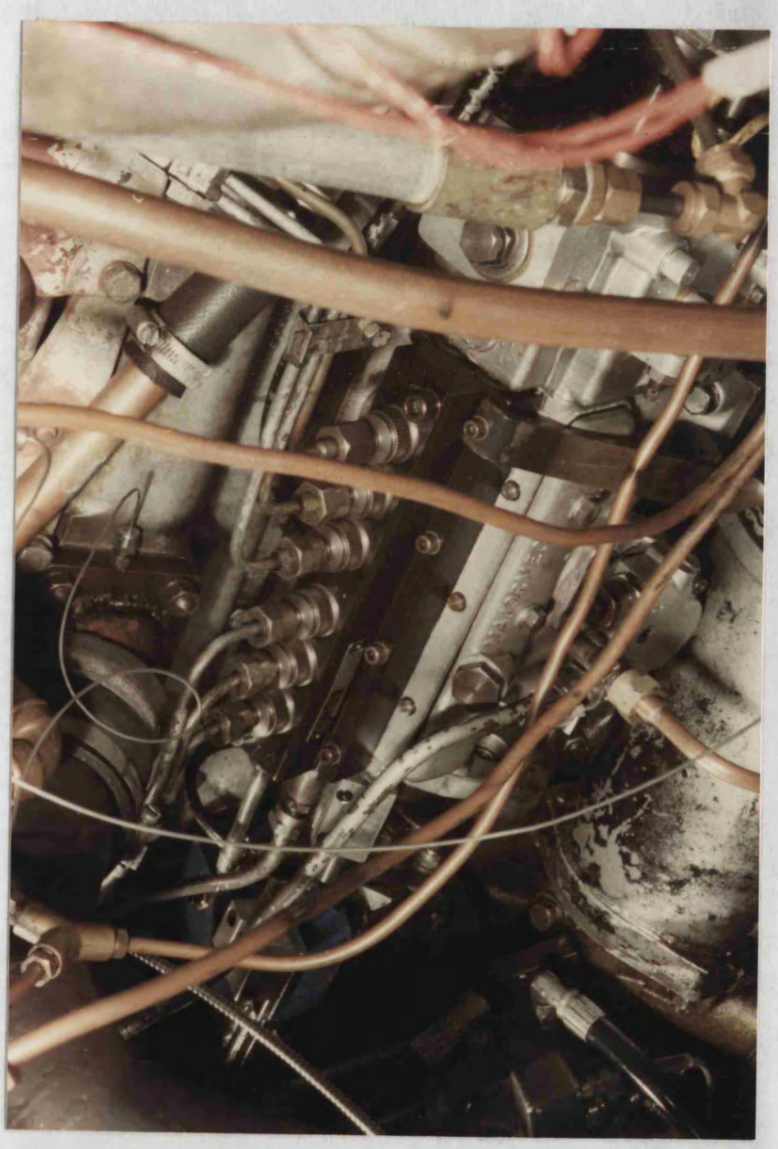


Figure 7.2. The Leyland TL11 engine fuel pump.

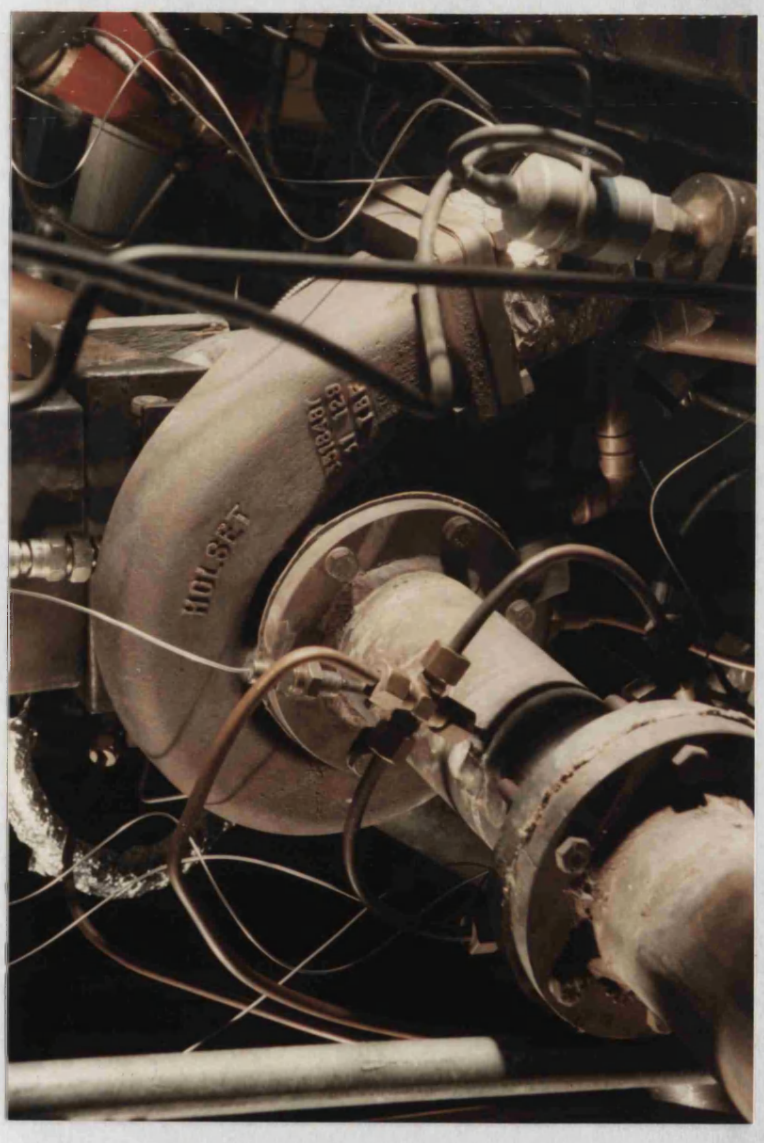


Figure 7.3. The Holset variable geometry turbocharger.

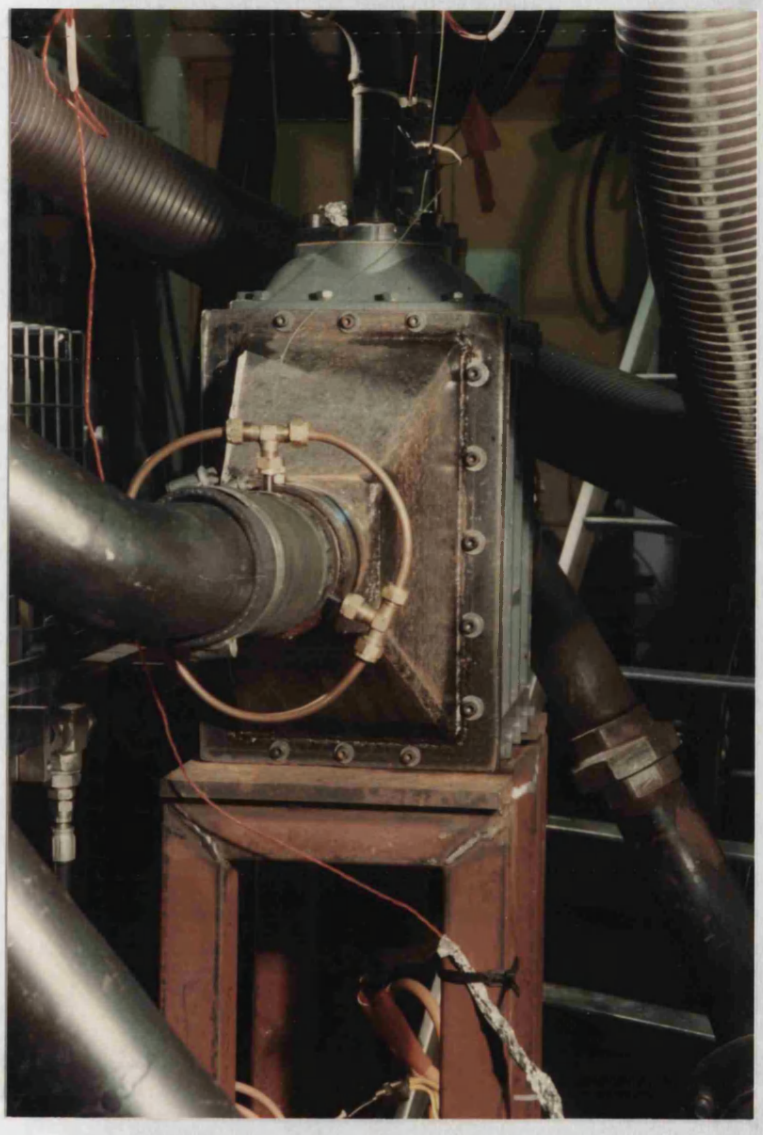


Figure 7.4. The Levland TL11 engine inter-cooler.

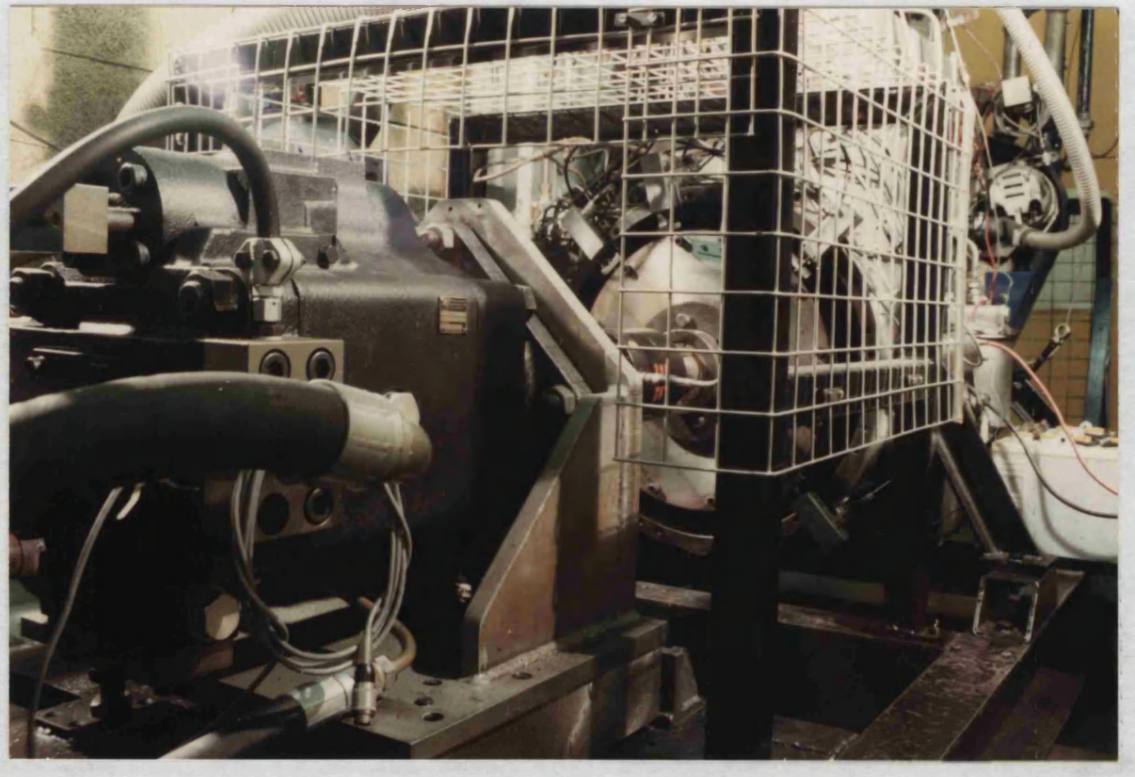


Figure 7.5. The flywheel end of the Leyland TL11 engine showing the hydraulic dynamometer.

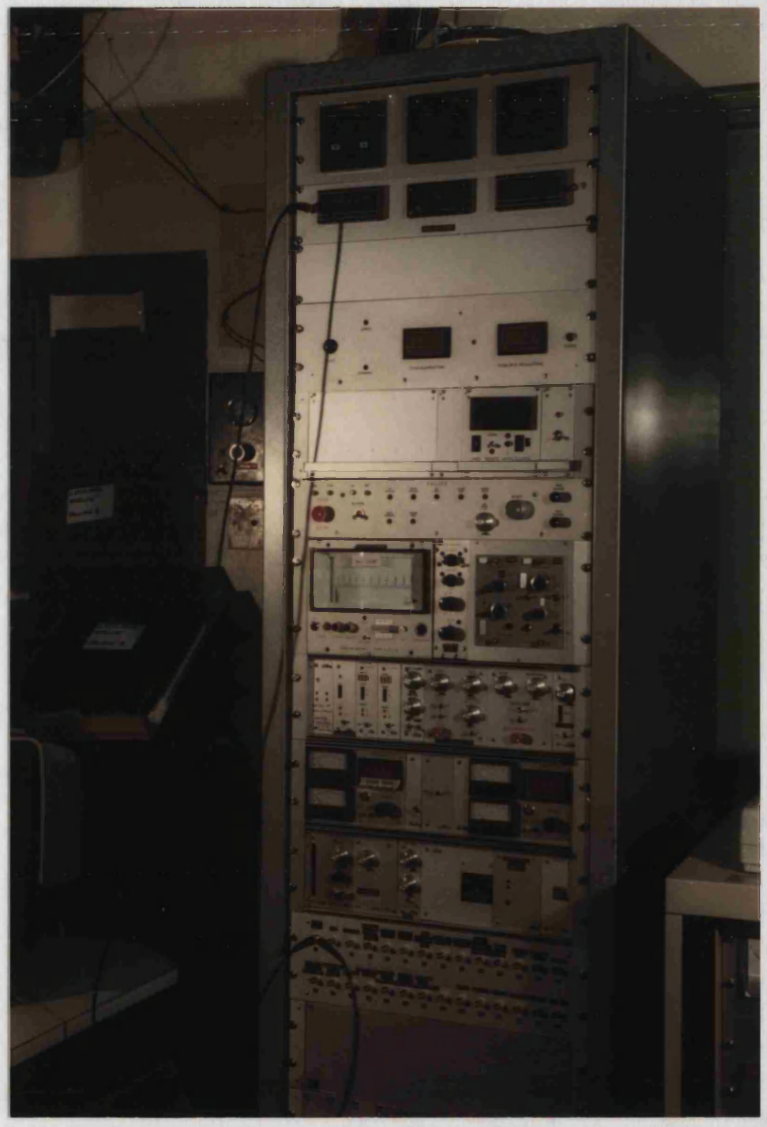


Figure 7.6. The Leyland TL11 engine instrumentation conditioning and control rack.

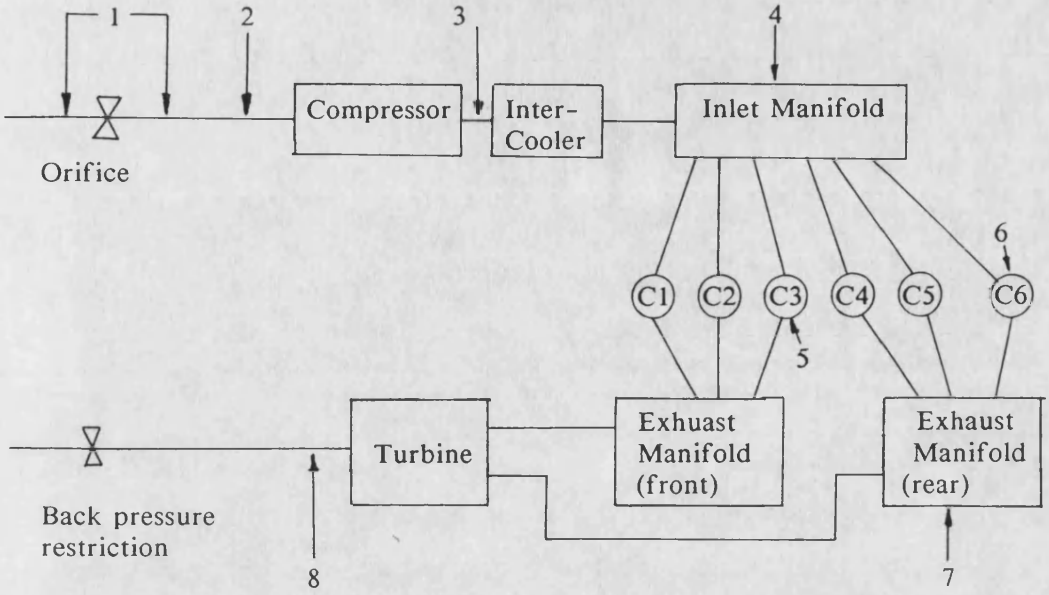


Figure 7.7 A schematic diagram of the Leyland TL11 engine pressure transducers. For details of the pressure transducers see text.

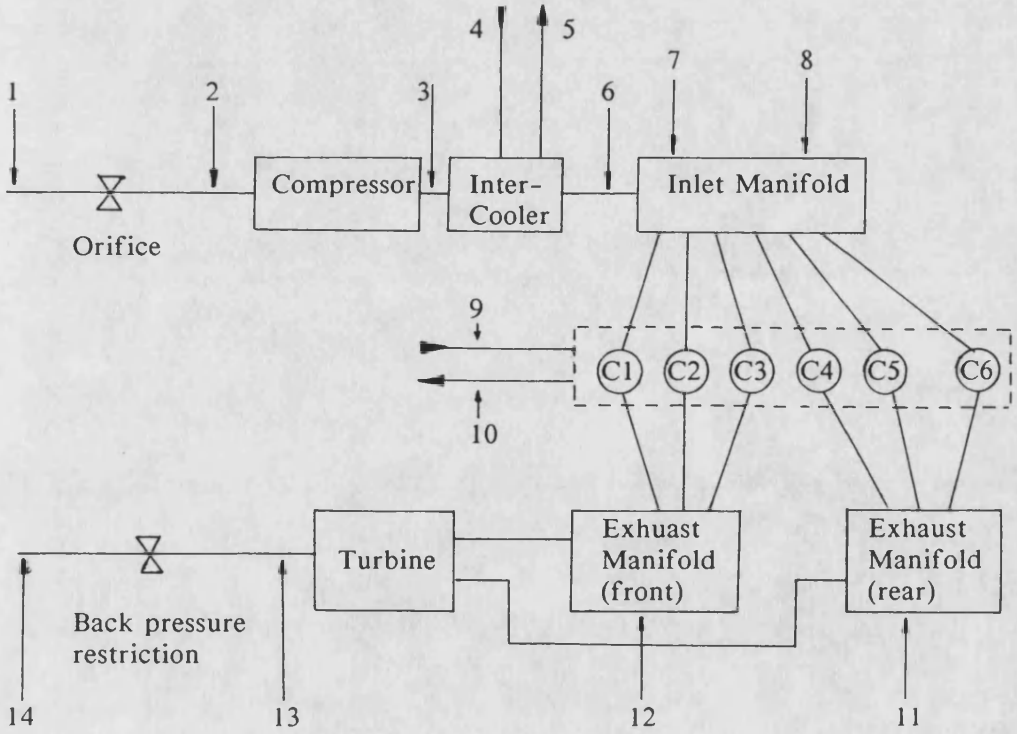


Figure 7.8 A schematic diagram of the Leyland TL11 engine temperature thermocouples. For details of the thermocouples see text.

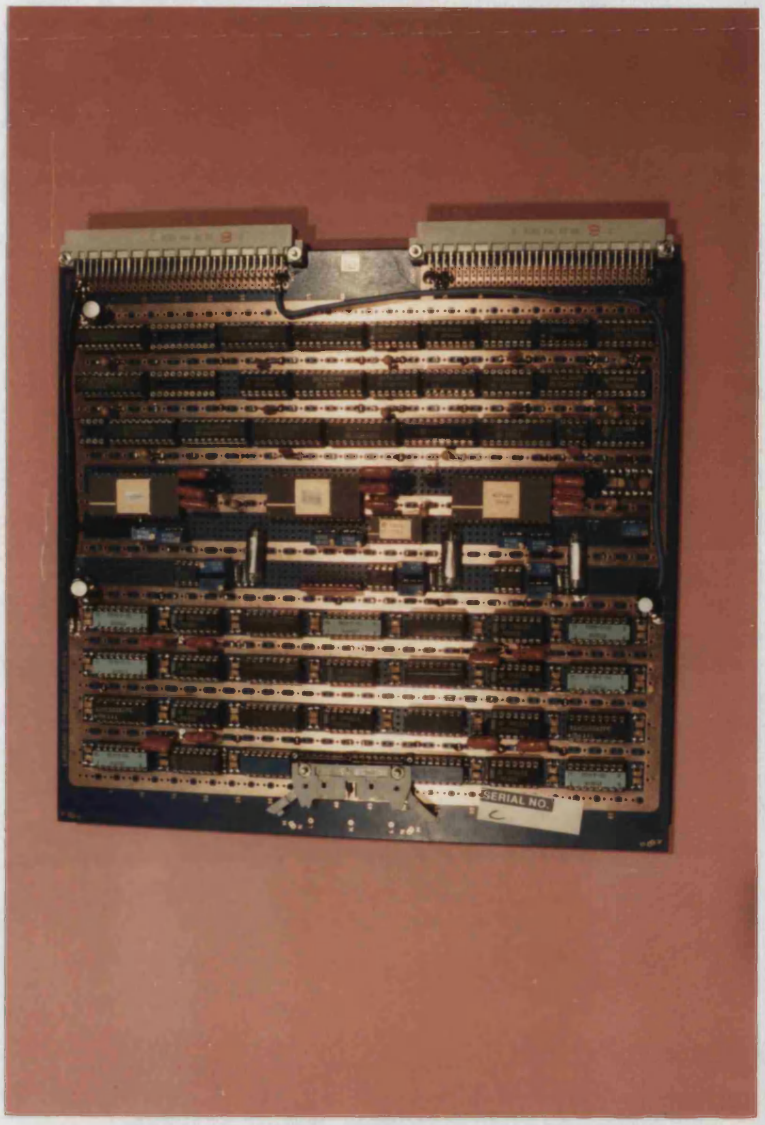


Figure 7.9 The analogue data acquisition board.

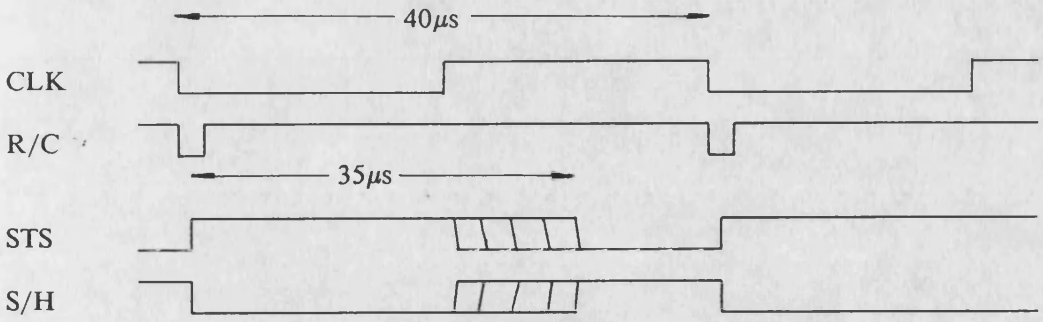


Figure 7.10 The timing diagram for the AD574 A/D conversion in free running mode.

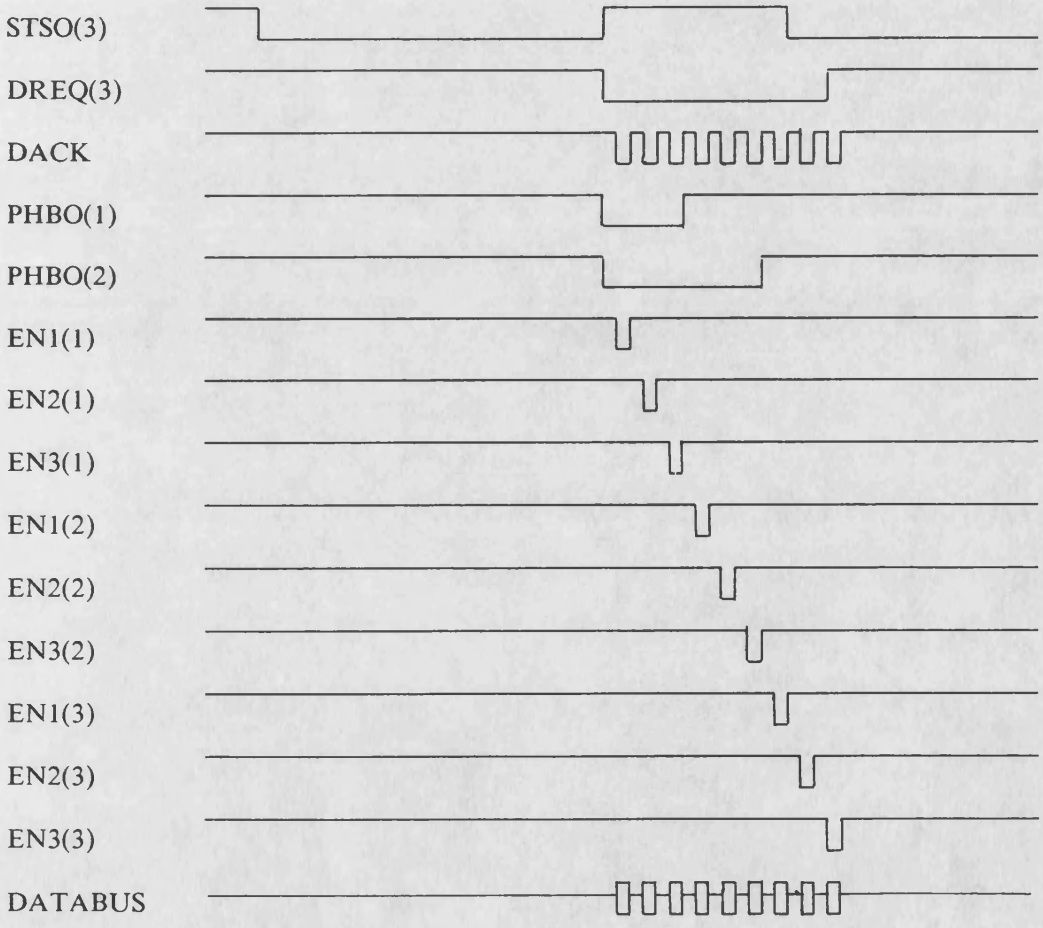


Figure 7.11 The DMA timing for three analogue data acquisition cards. The numbers in brackets indicate the board from which the signals originate.

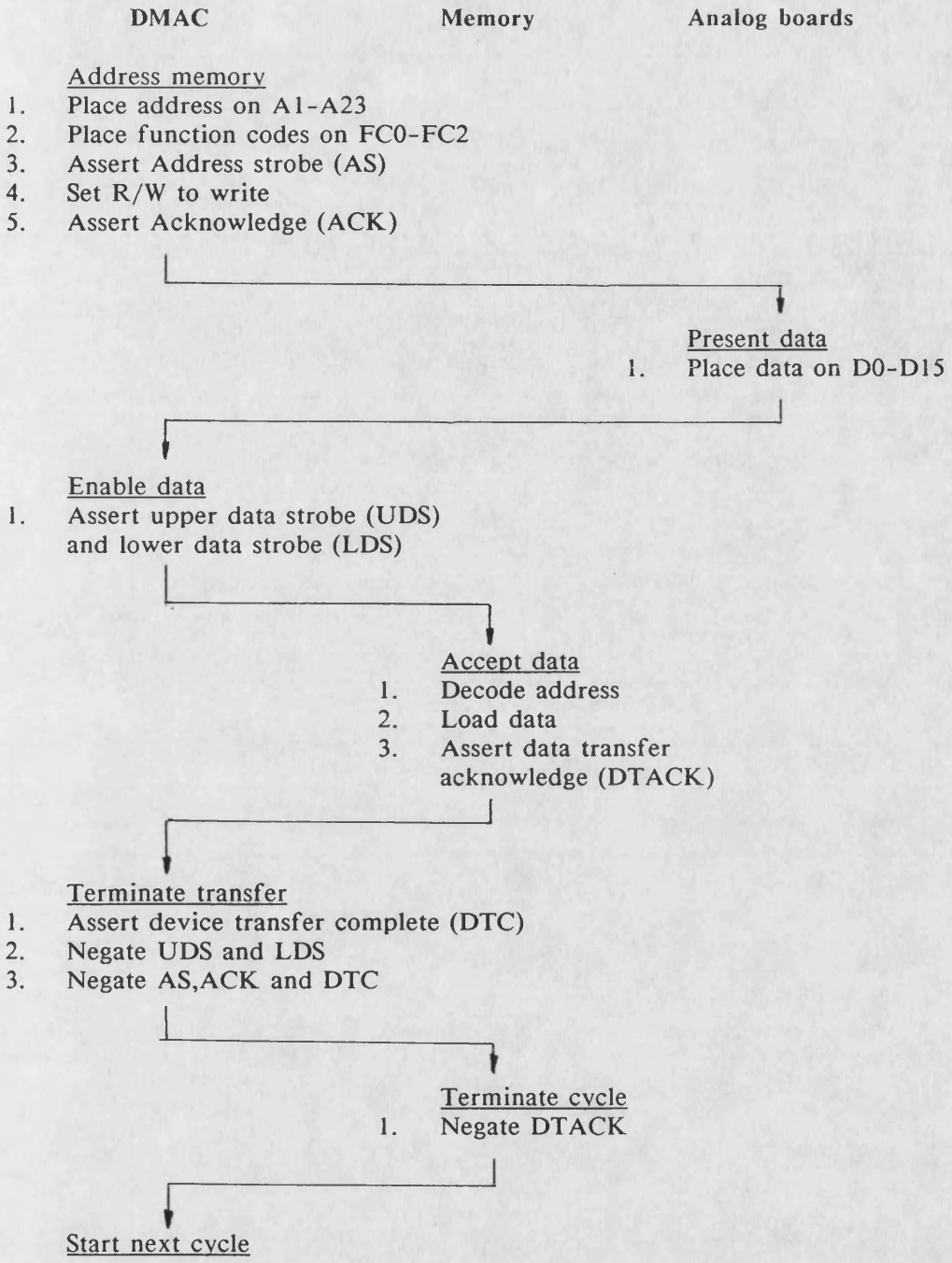


Figure 7.12 The sequence of actions performed during a single addressing mode transfer from the analogue boards to the memory.

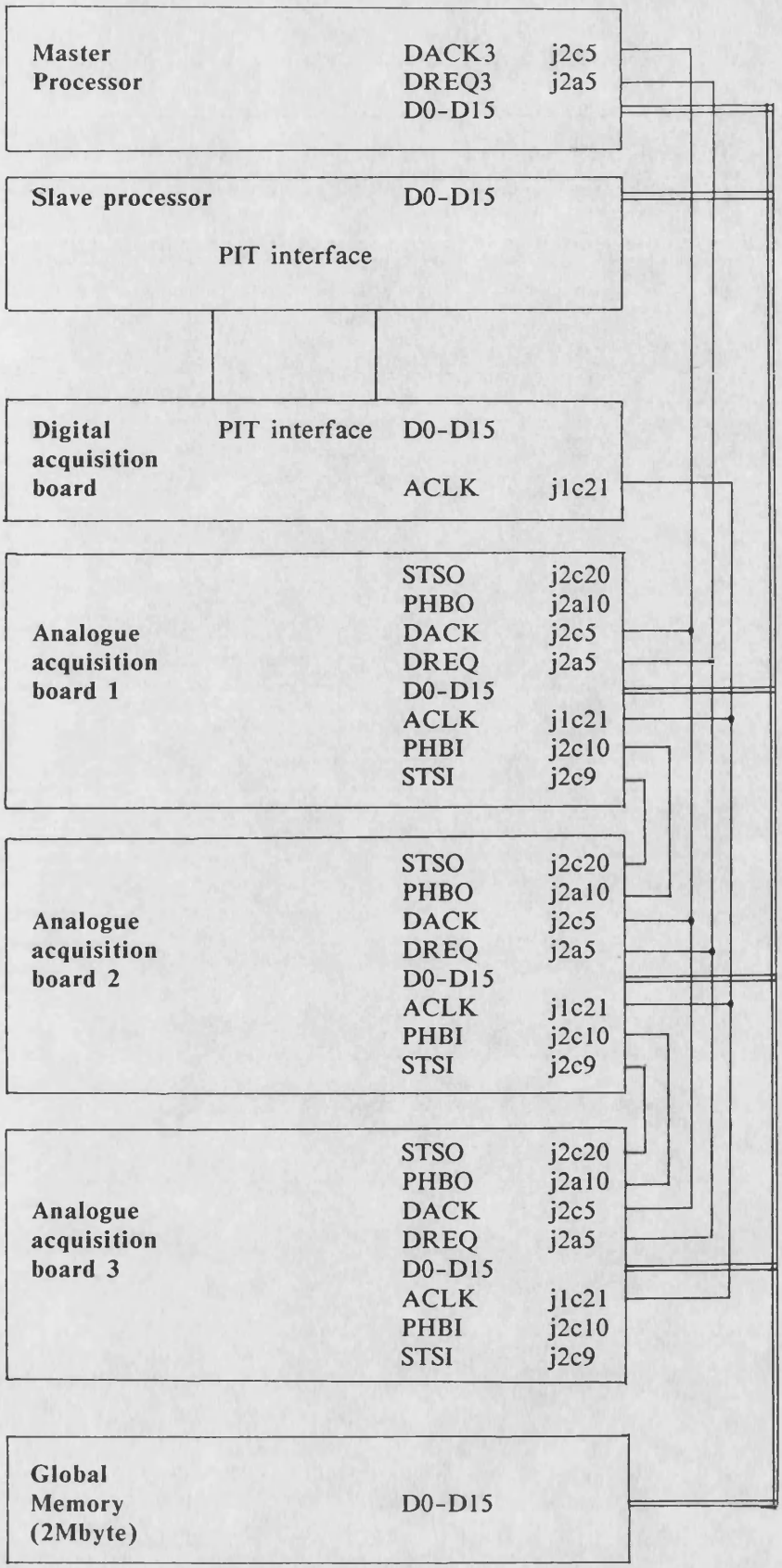


Figure 7.13. The backplane structure for the engine data acquisition computer.

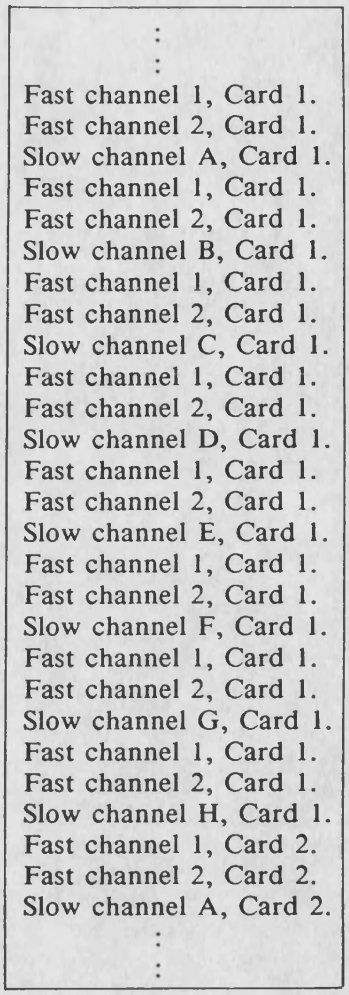


Figure 7.14 The analogue data memory structure.

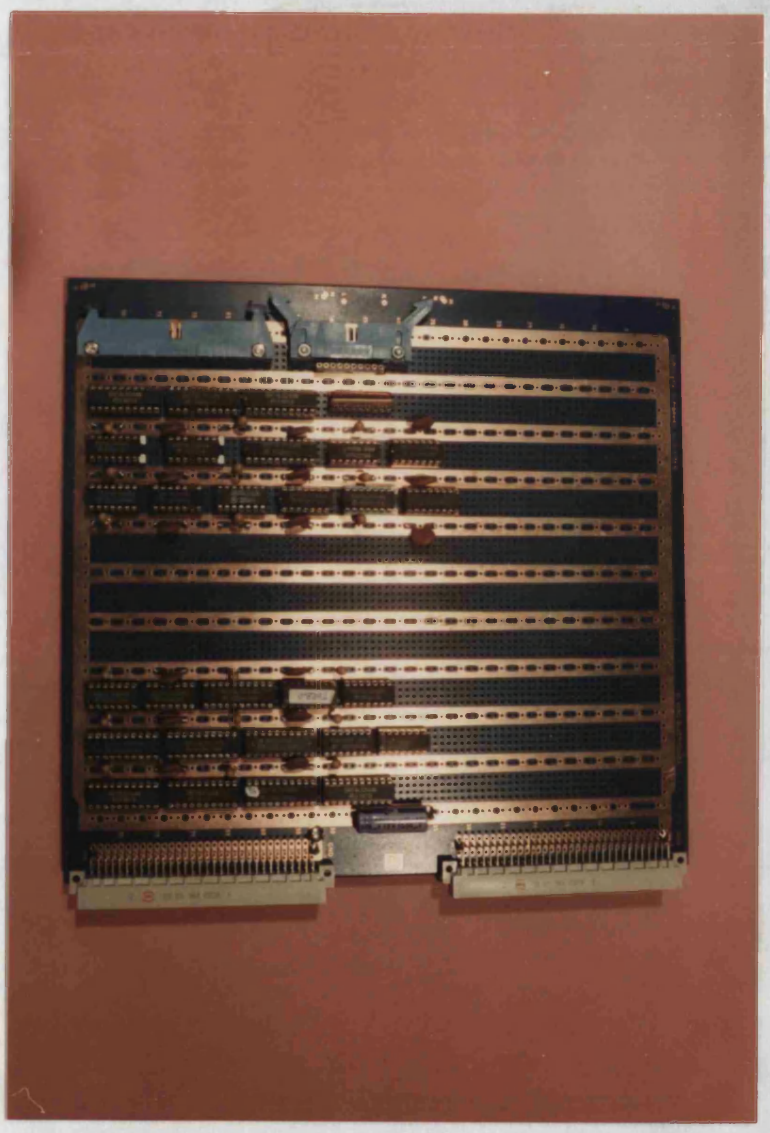
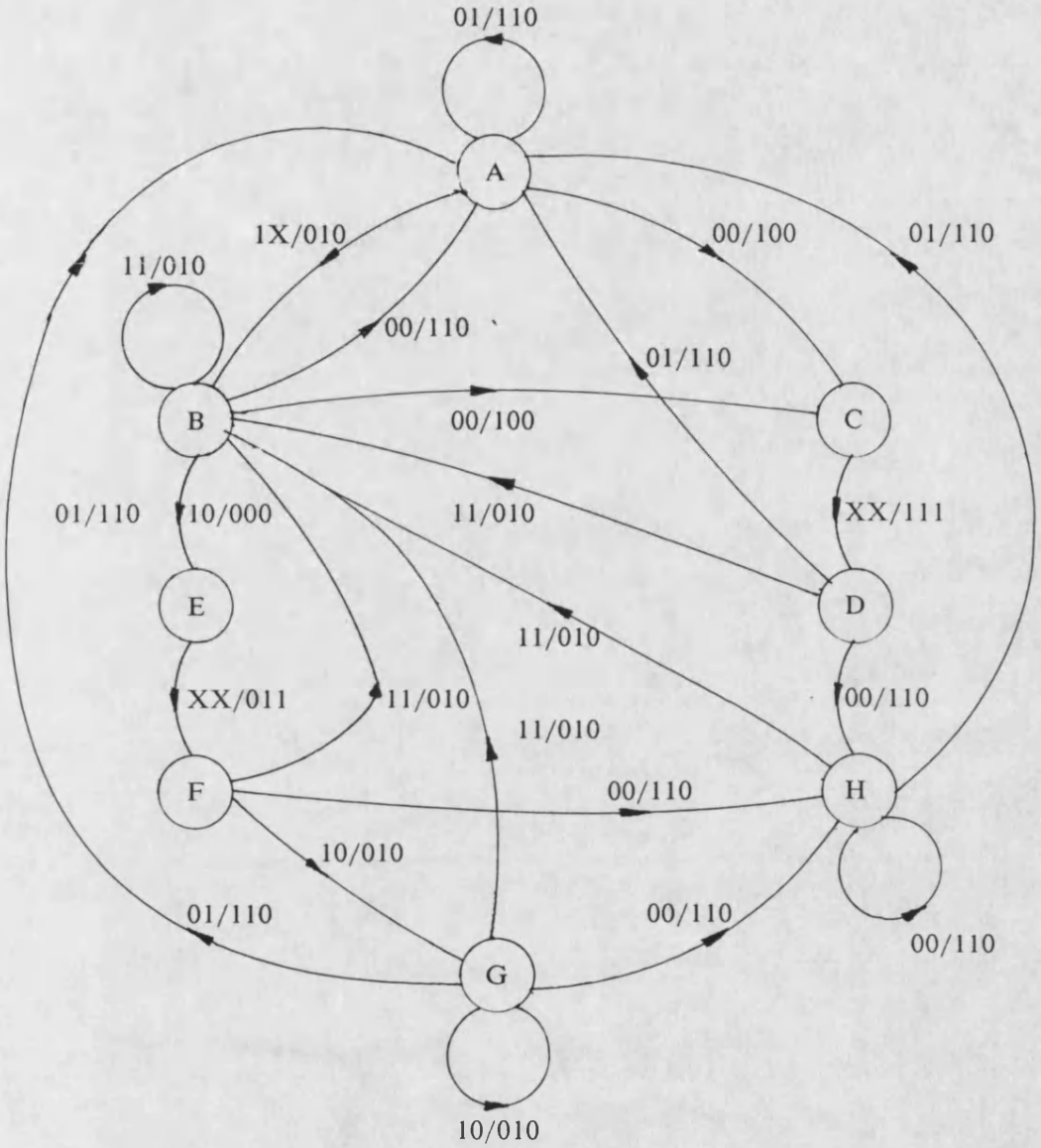


Figure 7.15 The digital data acquisition board.



Inputs/Outputs

Inputs = Turbine, 97.66Hz clock
Outputs = Counter clock, Interupt request, Clear counter

Figure 7.16. The digital acquisition board turbine counter state diagram.

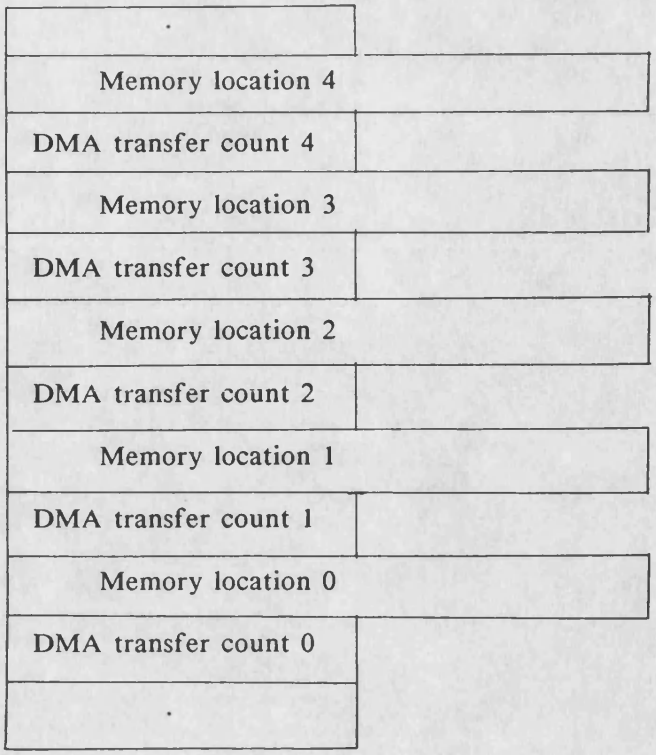


Figure 7.17a. The generalised memory structure for HD68450 DMA chained array memory transfers.

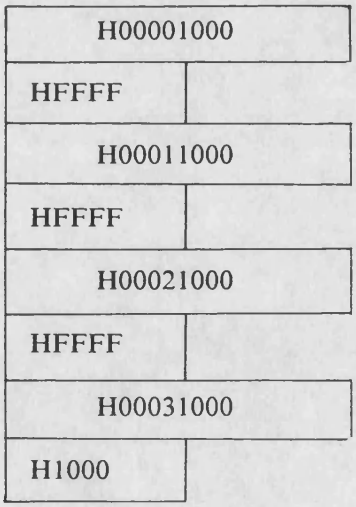


Figure 7.17b. The memory structure for a HD68450 DMA chained array memory transfer of length H31000 and starting at memory location H1000.

pkt.link
analogue device driver
action acquisition
result 1
result 2
BCPL pointer to result memory area
Length of result memory area in BCPL words.

Figure 7.18 The analogue acquisition driver packet structure.

pkt.link
digital device driver
action acquisition
result 1
result 2
BCPL pointer to angle result memory area
Length of result memory area in BCPL words.
BCPL pointer to turbine result memory area
Trigger.

Trigger = 0: external trigger enable.

Trigger = negative: internal trigger without engine running.

Trigger = positive: internal trigger with engine running

Figure 7.19 The digital acquisition driver packet structure.

H840040	ADC board trigger.
H840041	ADC board reset.
	:
	:
	:
H840050	Turbine counter location
	:
	:
	:
H840060	

Figure 7.20 The memory map for the data acquisition computer analogue and digital cards.

CHAPTER 8.

THE VALIDATION OF THE PARALLEL ENGINE SIMULATION.

8.1. Introduction.

For the purposes of validation the new parallel engine simulation, described in chapter 4, has been compared with an equivalent serial engine simulation and with results from a real engine. These comparisons are described in the following sections.

8.2. Comparison of the parallel engine simulation with an equivalent serial computer engine simulation.

The results from the DISC parallel diesel engine simulation have been compared to those for a serial computer diesel engine simulation. A serial engine simulation has been developed by White [61] and this is coded in ANSI 'C'. The simulation runs on an IBM PC compatible computer with an Intel 80386 processor. The serial simulation uses the Euler predictor corrector method of integration with a stability criterion of 0.1%. It has been verified against SPICE the serial engine simulation program developed by Charlton [16].

The serial simulation developed by White has no facility for variable wall temperature in the cylinder control volumes, neither does it calculate heat transfer in the manifold control volumes. To allow a comparison between the simulations, the parallel simulation was modified to make the calculation of cylinder wall temperature and the calculation of manifold heat transfer conditionally compilable. The serial simulation was initialised to use the Hohenburg [11] calculation for heat transfer coefficient and to use the same effective valve area tables as the parallel simulation.

The results from both simulations are given in tables 8.1 and 8.2 and the two sets of results are nearly identical. The calculated values of turbine speed are slightly different. However this can be explained by the different mapping of the compressor performance map (figure 2.6) in the two simulations. The difference in turbine speeds is small when superimposed onto the compressor map. The value for Pumping Mean Effective Pressure (PMEP) is also different. This is because the two simulations calculate PMEP differently. The parallel simulation calculates PMEP from exhaust valve open to inlet valve close. The serial simulation calculates PMEP for the open half of the engine cycle from top dead centre to top dead centre.

Table 8.1. The results from the parallel DISC diesel engine simulation.

Cylinder information

IMEP	(bar)	14.064			
PMEP	(bar)	0.10953	Exhaust Energy	(J)	2037.2
FMEP	(bar)	1.9886	Fuel Energy	(J)	5476.3
BMEP	(bar)	12.075	Output Work	(J)	2236.9
Mechanical Efficiency		85.86%	Friction Work	(J)	368.38
Indicated power	(kW)	195.4	Wall Energy	(J)	826.35
Brake power	(kW)	167.77	Energy Balance		1.0014
ISFC	(g/kWhr)	176.81	Inlet Mass	(g)	3.2154
BSFC	(g/kwhr)	205.92	Fuel Mass	(g)	0.12795
Indicated Efficiency		47.573%	Exhaust Mass	(g)	3.3382
Brake Efficiency		40.847%	Mass Balance		1.0015
Volumetric Efficiency		95.107%	Engine Speed	(rpm)	1500
Fuel flow	(g/s)	9.5963	Maximum Pressure	(bar)	133.54

Manifold information

Turbine speed	(rpm)	71163	Compressor Efficiency		75.62%
In. man pressure	(bar)	1.6221	In. man temperature	(K)	304.29
Ex1 man pressure	(bar)	1.7656	Ex1 man temperature	(K)	813.84
Ex2 man pressure	(bar)	1.7654	Ex2 man temperature	(K)	813.42
Compressor power	(kW)	-17.09	Turbine1 power	(kW)	8.528
Turbine efficiency		50%	Turbine2 power	(kW)	8.522
In. man mass in	(g)	19.266	In. man mass out	(g)	19.231
Ex1 man mass in	(g)	10.012	Ex1 man mass out	(g)	10.016
Ex2 man mass in	(g)	10.014	Ex2 man mass out	(g)	10.018
In. man Energy in	(J)	5804.6	In. man Energy out	(J)	5793.7
Ex1 man Energy in	(J)	9027.3	Ex1 man Energy out	(J)	9030.1
Ex2 man Energy in	(J)	9023.5	Ex2 man Energy out	(J)	9026.4
In. man mass balance		1.0018	In. man Energy balance		1.0019
Ex1 man mass balance		0.99964	Ex1 man Energy balance		0.99968
Ex2 man mass balance		0.99964	Ex2 man Energy balance		0.99968

Table 8.2. The results for the White serial diesel engine simulation.

Cylinder information

IMEP	(bar)	14.1
PMEP	(bar)	-0.23
FMEP	(bar)	1.99
BMEP	(bar)	12.1
Mechanical Efficiency		85.85%
Indicated power	(kW)	195.6
Brake power	(kW)	168.0
ISFC	(g/kWhr)	176.2
BSFC	(g/kwhr)	205.2
Brake Efficiency		40.99%
Volumetric Efficiency		95.7%
Fuel flow	(g/s)	9.5785
Maximum Pressure	(bar)	133.0

Manifold information

Turbine speed	(rpm)	73529.7	Compressor Efficiency		75.79%
In. man pressure	(bar)	1.648	In. man temperature (K)		305.0
Ex1 man pressure	(bar)	1.757	Ex1 man temperature(K)		803.0
Ex2 man pressure	(bar)	1.766	Ex2 man temperature(K)		800.0
Compressor power	(kW)	-17.85	Turbine1 power	(kW)	8.59
Turbine efficiency		50%	Turbine2 power	(kW)	8.75

System mass balance		System energy balance	*[J]*	*[%]*
Air in	0.01959659	Energy of fuel	32796.9	100.00
Fuel in	0.00076628	Energy to exhaust	10832.2	33.03
Exhaust out	0.02036373	Energy to coolant	4876.8	14.87
Mass balance	1.0000	Useful work	13445.0	40.99
		Friction work	2215.9	6.76
		Intercoolers	1419.6	4.33
		Shaft losses	41.6	0.13
		Energy balance	0.9990	

8.3. The comparison of the parallel engine simulation with the TL11 engine.

8.3.1. The comparison of the engine and the simulation for steady state operating points across the engine torque range.

The parallel simulation was compared with the TL11 engine described in chapter 7. The steady state operating parameters of the engine and the simulation were compared at two engine speed points for the complete engine torque range. The two speeds that were used are 1500rpm and 2000rpm.

For both sets of results, the engine was run with the dynamometer in its constant torque mode. The following procedure was used for each set of steady state results taken from the experimental engine:

- The engine fuel rack input was incremented.
- This caused an increase in engine speed, as the demanded engine torque remained constant.
- The demanded torque was increased to bring the engine speed back to the desired value.
- The results were then taken from the engine after a delay to allow the engine to reach a steady state condition.

For the comparison with the results from the engine, the parallel simulation was set up to give a constant engine speed. This was achieved by giving the simulated engine a large load inertia. With the simulation running at constant speed, the fuel rack input was adjusted to give the same fuel flow rates as had been obtained from the engine. For each fuel flow rate point the simulation was

allowed to reach a steady state before any results were taken. The simulation was judged to have reached a steady state when the mass and energy balance for the cylinder control volume were within 0.5% of perfect balance. The simulation was run with an initial integration step length of 0.5^{o} and with a stability criterion of 0.1%.

A comparison of the results from the engine and the simulation are shown in figures 8.1 for an engine speed of 1500 rpm and in figures 8.2 for an engine speed of 2000 rpm. In both sets of figures the results from the engine and the simulation are plotted against engine fuel flow rate. The results from the engine are indicated by circles and the results from the simulation by crosses. The turbine speed measurements from the engine could not be obtained for the 2000 rpm tests due to the mal-functioning of the turbine speed transducer.

The graphs for engine BMEP figures 8.1a and 8.2a have had a Willands line [3] superimposed on them. Using this line, a rough approximation of the Friction Mean Effective Pressure (FMEP) can be obtained. The engine FMEP is estimated as the negative value of BMEP at zero output torque (zero fuel flow rate). The graphs of BMEP against fuel flow rate give values of FMEP of 1.5 bar and 2 bar for the engine speeds of 1500 rpm and 2000 rpm respectively.

The graphs for the engine performance parameters BMEP, BSFC, Brake power and engine air flow all show the simulation indicating the same trends as the engine. This match between the simulation and the engine should be improved by adjusting the empirical models used in the simulation with results from the engine. For example, the graphs for inlet manifold temperature indicate that the intercooler model used in the parallel simulation is too simple. With a constant for intercooler effectiveness, the simulated inlet manifold temperature virtually

remains constant whereas the effectiveness in fact varies depending on the air flow through the intercooler.

8.3.2. The comparison of the TL11 engine and the simulation for different fuel injection timings.

On the experimental engine it is possible to adjust the fuel injection timing. A comparison was therefore made between the engine and the simulation for different fuel injection timings. The fuel injection timing of the experimental engine is voltage controlled from the engine instrumentation rack. The control determines the static injection timing of the engine with a value of 0.7V giving the standard injection timing of 12^{o} before TDC. The engine was run at various fuel injection timing inputs from 0 to 10 V. The high speed in-cylinder pressure and needle lift outputs from the engine were recorded for each input as well as the slow speed data used in section 8.3. For each set of results the engine was run with the dynamometer in its constant torque mode, with a torque of 500Nm. The injection timing was adjusted and the fuel rack was then adjusted to bring the engine speed back to the desired speed of 1500 rpm.

The Kistler pressure transducers have an output dc offset which varies with operating condition. The results from the transducers must therefore be calibrated to a known value of pressure. For these experiments, the value of pressure in the cylinder during scavenge was taken as the value of inlet manifold pressure.

The relationship between the fuel injection input and the static injection timing could not be easily determined. Therefore using the results for needle lift, which indicate when fuel is injected into the cylinder, the dynamic fuel injection timings were determined. The traces for needle lift against crankshaft angle for the injection inputs 1V, 3V, 5V, 7V, 9V and 10V are shown in figure 8.3. The dynamic injection timings determined from these traces are given in table 8.3.

Table 8.3. The variation of dynamic injection timing with respect to injection timing control input for an engine speed of 1500 rpm.

Control input.	Dynamic injection timing. (w.r.t. TDC)	
0V	368.6^{o}	$(+8.6^{o})$
1V	368.6^{o}	$(+8.6^{o})$
3V	363.8^{o}	$(+3.8^{o})$
5V	359.4^{o}	(-0.6^{o})
7V	354.2^{o}	(-5.8^{o})
9V	350.0^{o}	(-10.0^{o})
10V	347.4^{o}	(-12.6^{o})

The parallel simulation is normally controlled by a user input for static injection timing. The dynamic injection timing is determined from the static timing and a time delay which models the flow of the fuel down the injection pipe from the fuel pump. For these experiments, the modelled time delay was removed from the calculation of injection timing. Instead, the simulation was controlled by the dynamic injection timings obtained from the needle lift transducers on the experimental engine. The simulation was run with a large load inertia and a

constant speed of 1500rpm. The dynamic timing and the fuel flow were adjusted to match the results from the experimental engine. The simulation was then allowed to reach a steady state before a data log was taken of the in-cylinder parameters with respect to crankshaft angle.

Figures 8.4 show the results for the simulation and the experimental engine for the different fuel injection timings. The results show that the simulation models the combustion process in the cylinder fairly well. Some of the models used in the parallel simulation have empirical factors which may be adjusted to match specific engines. For the parallel simulation, these factors have been set to the default parameters suggested by the authors of the models. The model which most notably could be improved is that for ignition delay. The Wolfer model [12] appears to underestimate ignition delay for retarded injection timings.

8.4. Conclusions.

The parallel simulation has been compared to both an equivalent serial simulation and to results from an experimental engine. The results from the present model shows good correspondence with the results from the engine. However better matching between the engine and the simulation will be possible with the use of improved empirical models based on experimental results from the TL11 engine.

Figure 8.1a BMEP against fuel flow at 1500rpm for the parallel simulation (+) and the TL11 engine (o).

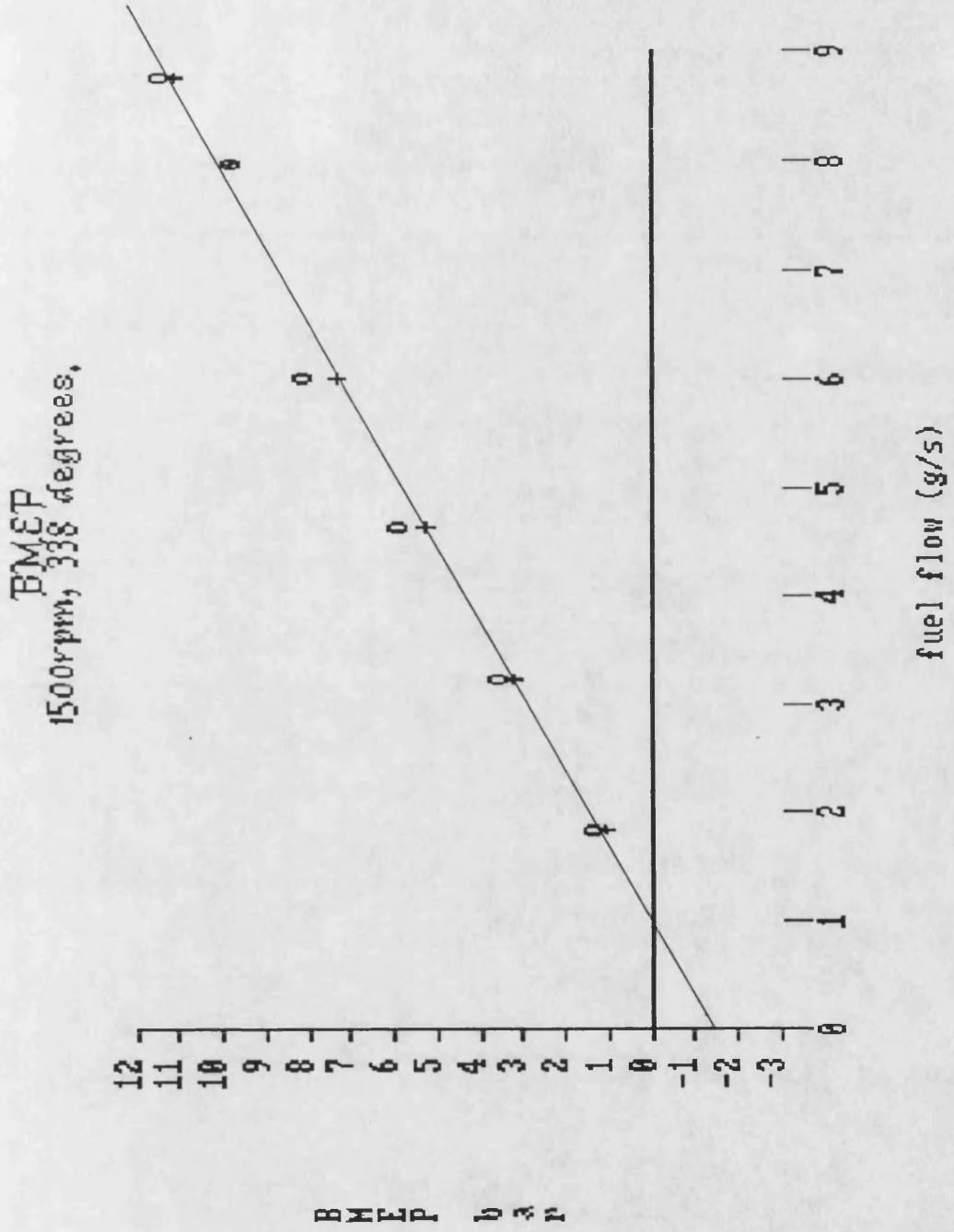


Figure 8.1b Brake Power against fuel flow at 1500rpm for the parallel simulation (+) and the TL11 engine (o).

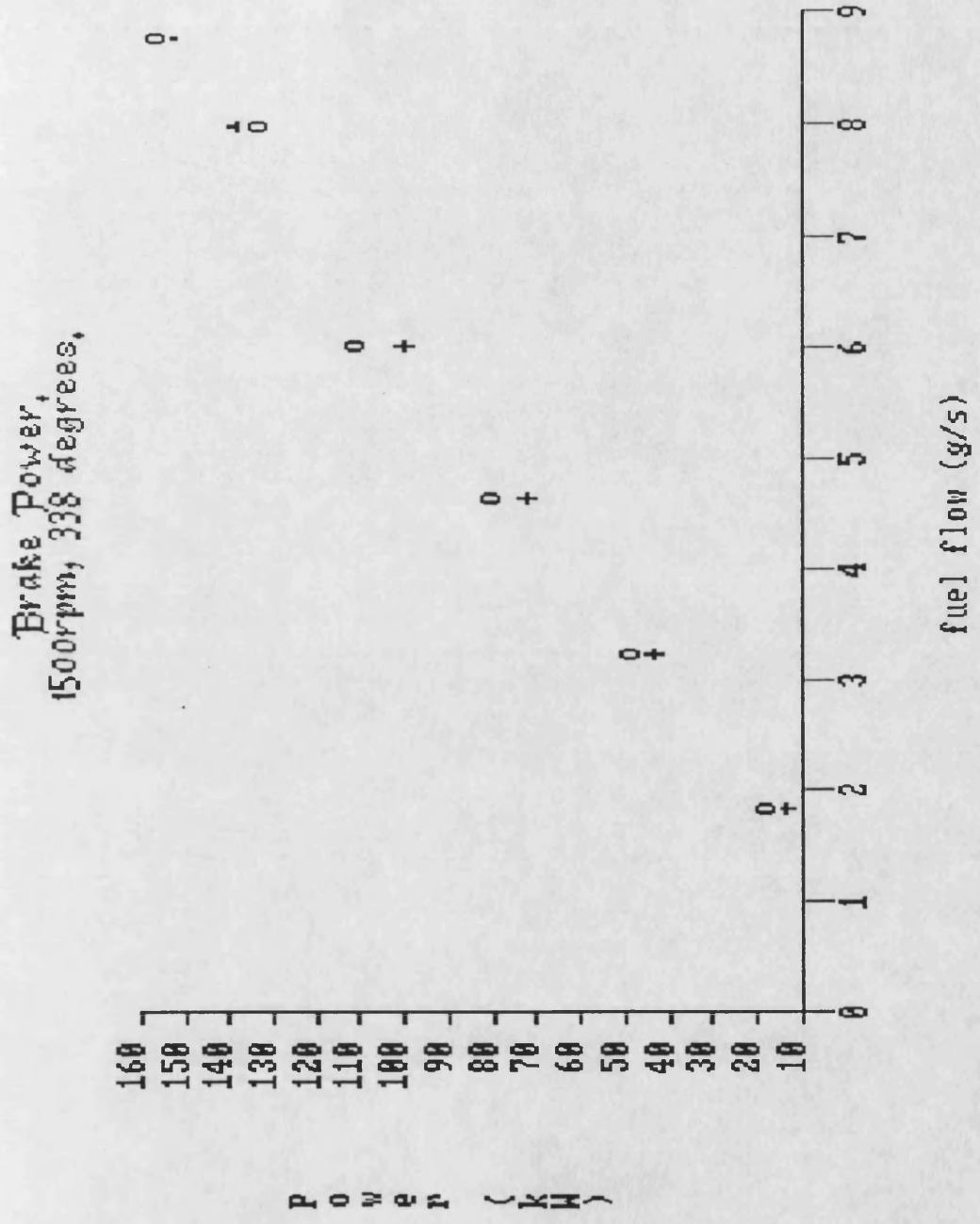


Figure 8.1c Brake efficiency against fuel flow at 1500rpm for the parallel simulation (+) and the TL11 engine (o).

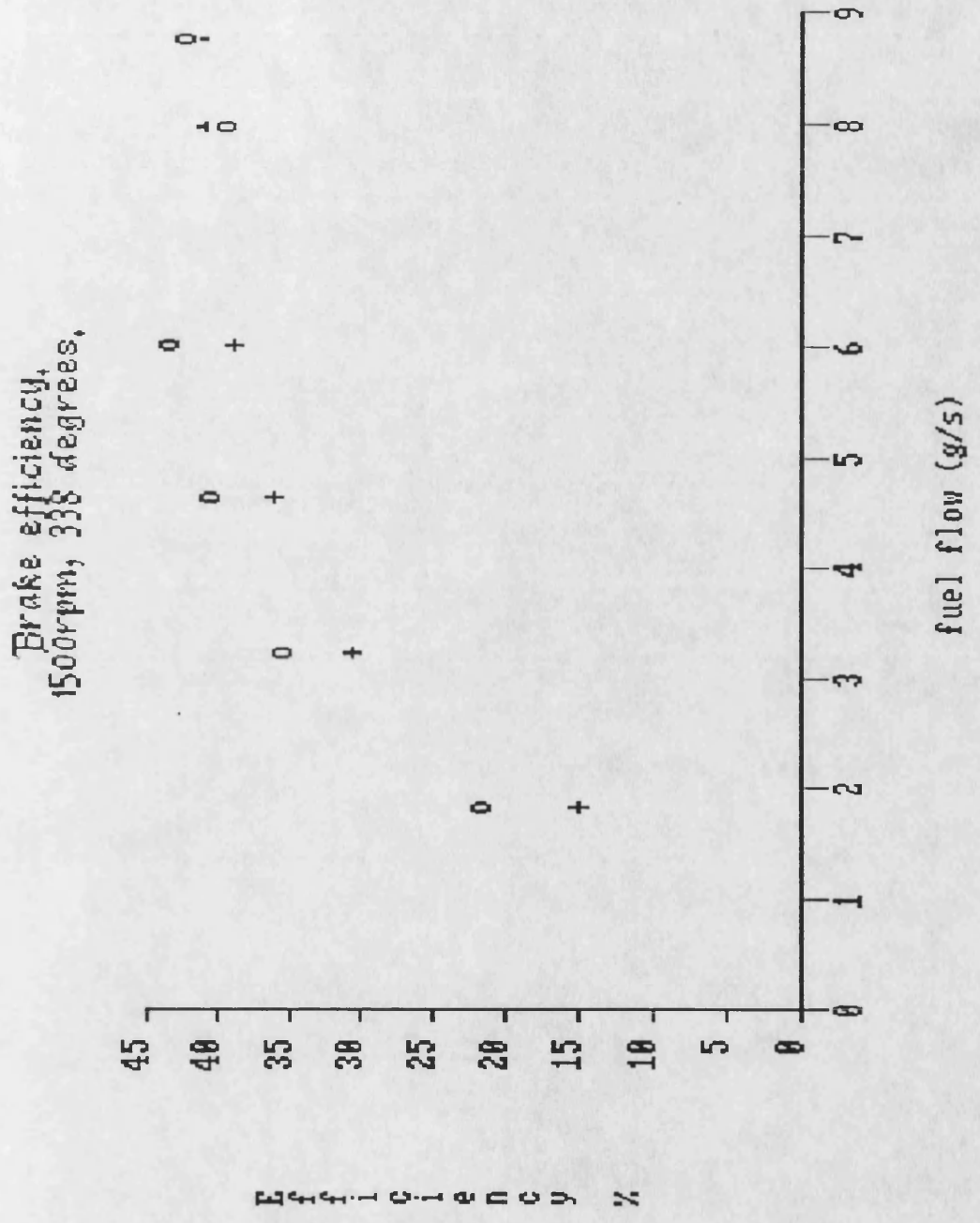


Figure 8.1d BSFC against fuel flow at 1500rpm for the parallel simulation (+) and the TL11 engine (o).

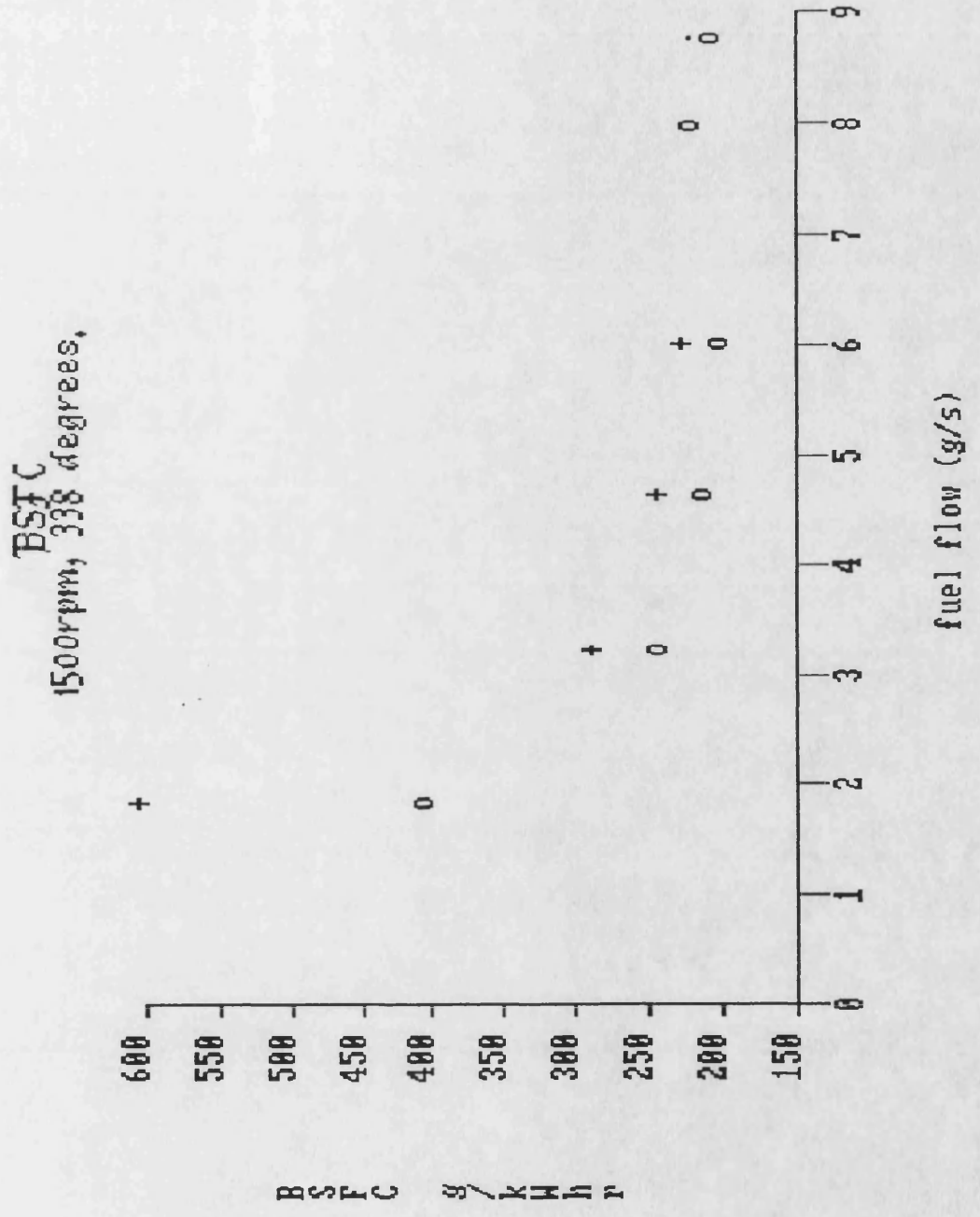


Figure 8.1e Air flow against fuel flow at 1500rpm for the parallel simulation (+) and the TL11 engine (o).

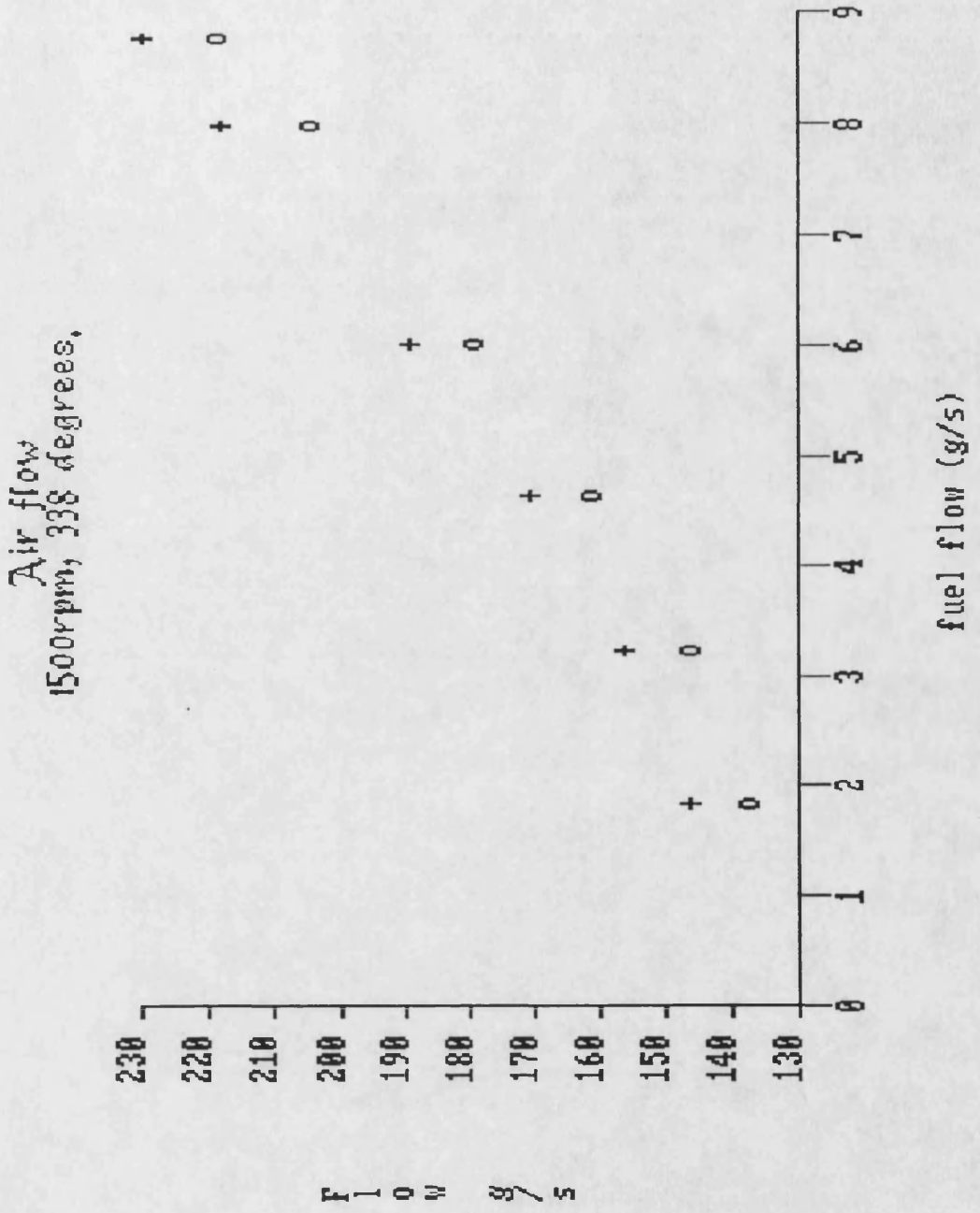


Figure 8.1f Inlet manifold temperature against fuel flow at 1500rpm for the parallel simulation (+) and the TL11 engine (o).

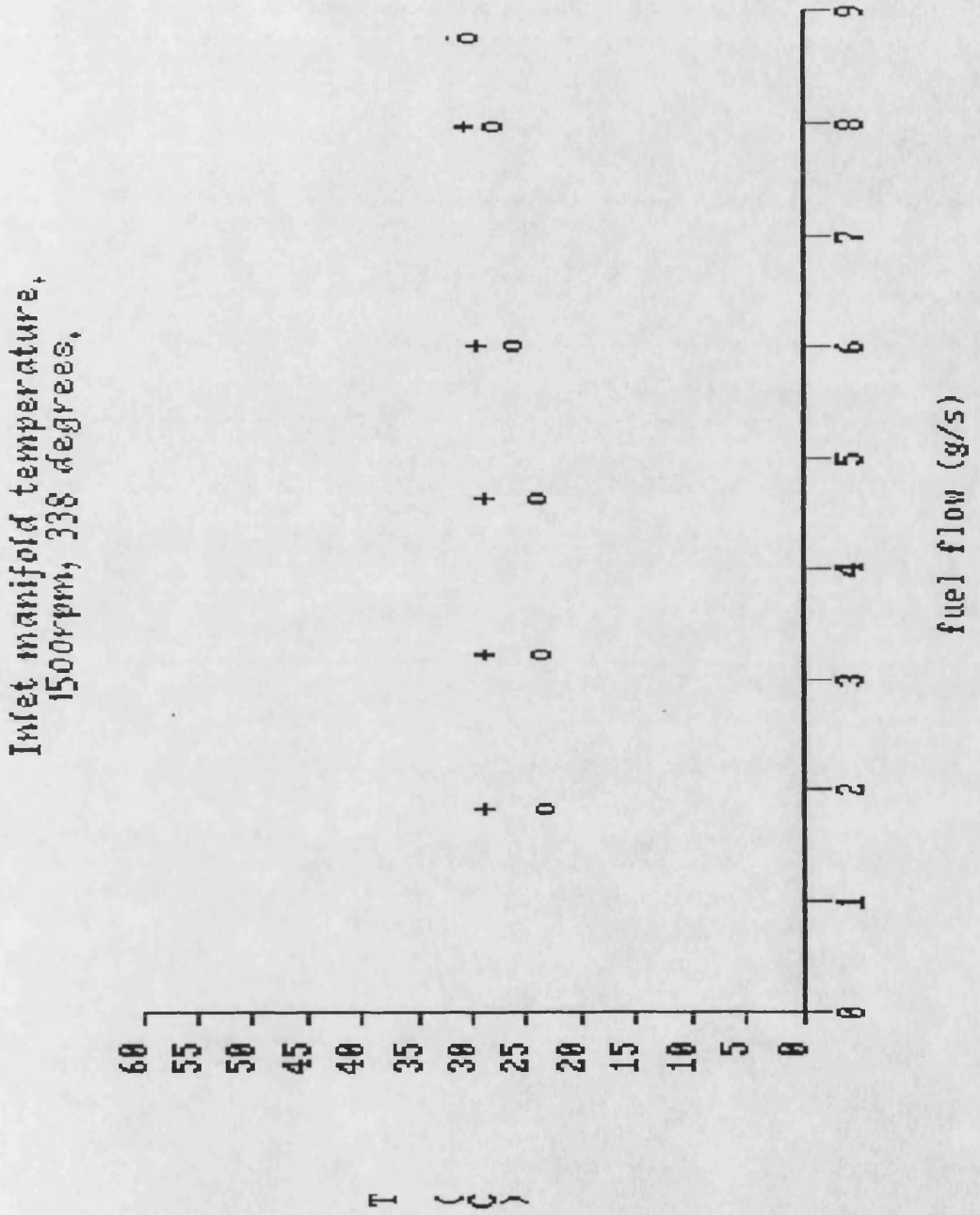


Figure 8.1g Exhaust manifold temperature against fuel flow at 1500rpm for the parallel simulation (+) and the TL11 engine (o).

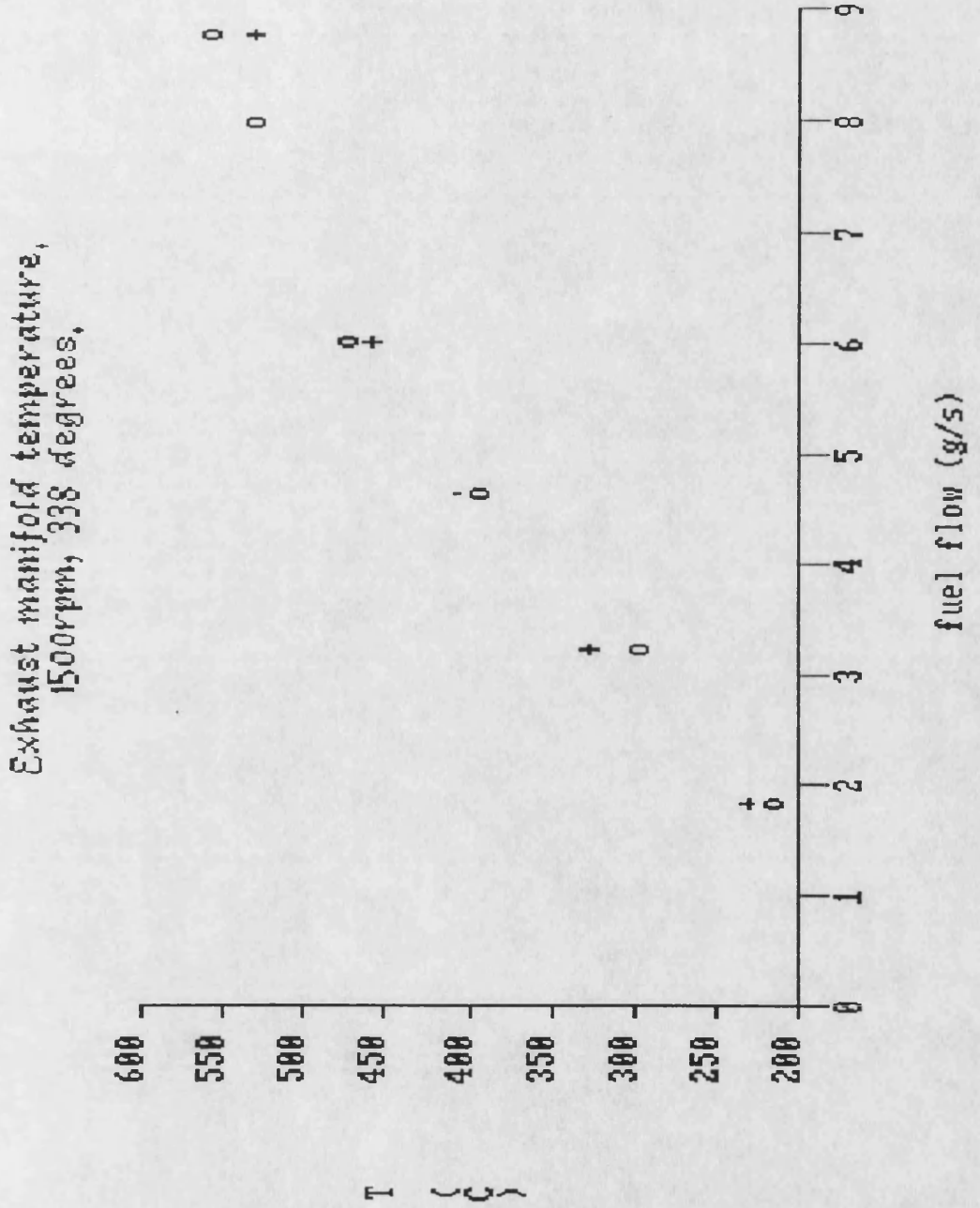


Figure 8.1h Inlet manifold pressure against fuel flow at 1500rpm for the parallel simulation (+) and the TL11 engine (o).

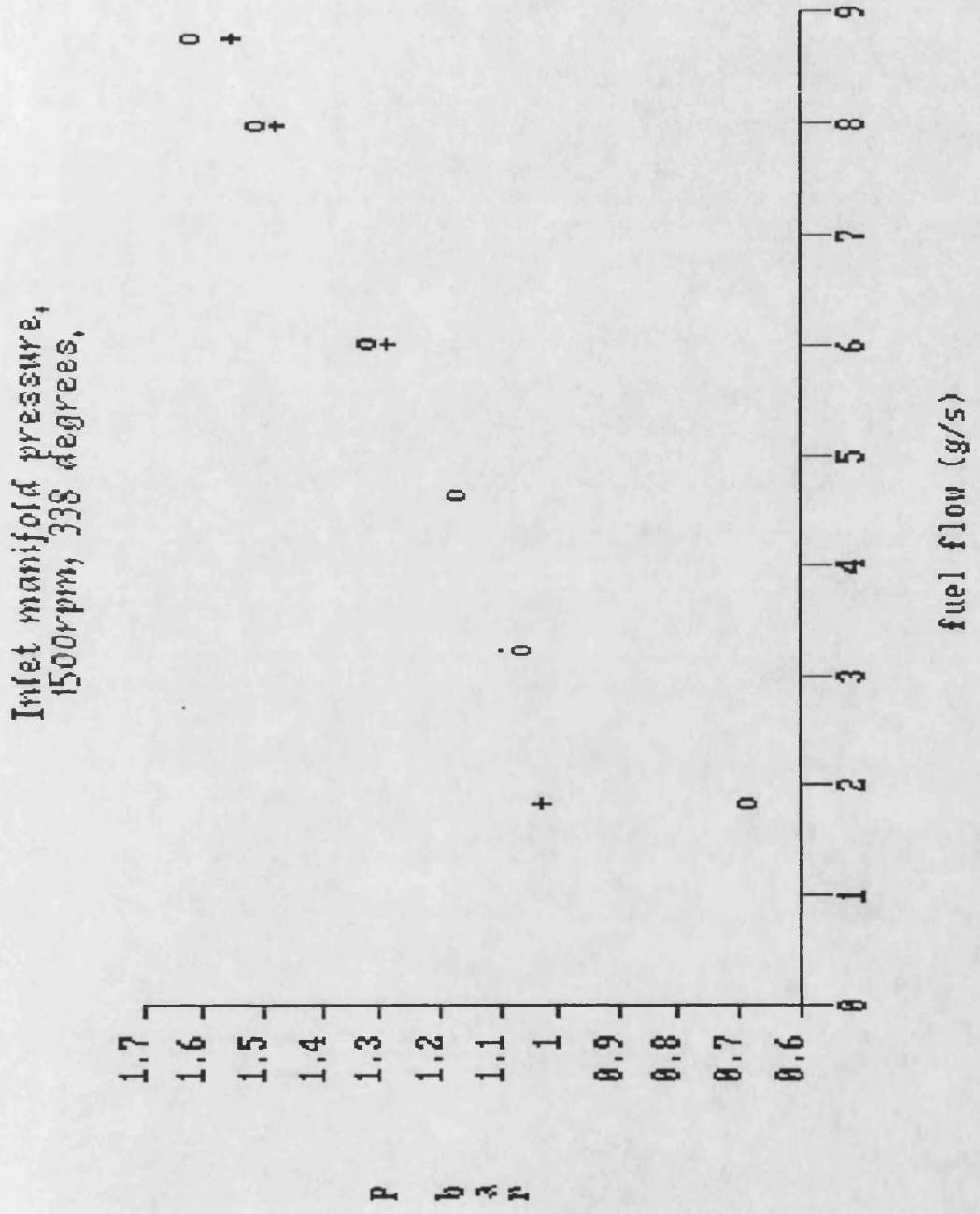


Figure 8.1i Exhaust manifold pressure against fuel flow at 1500rpm for the parallel simulation (+) and the TL11 engine (o).

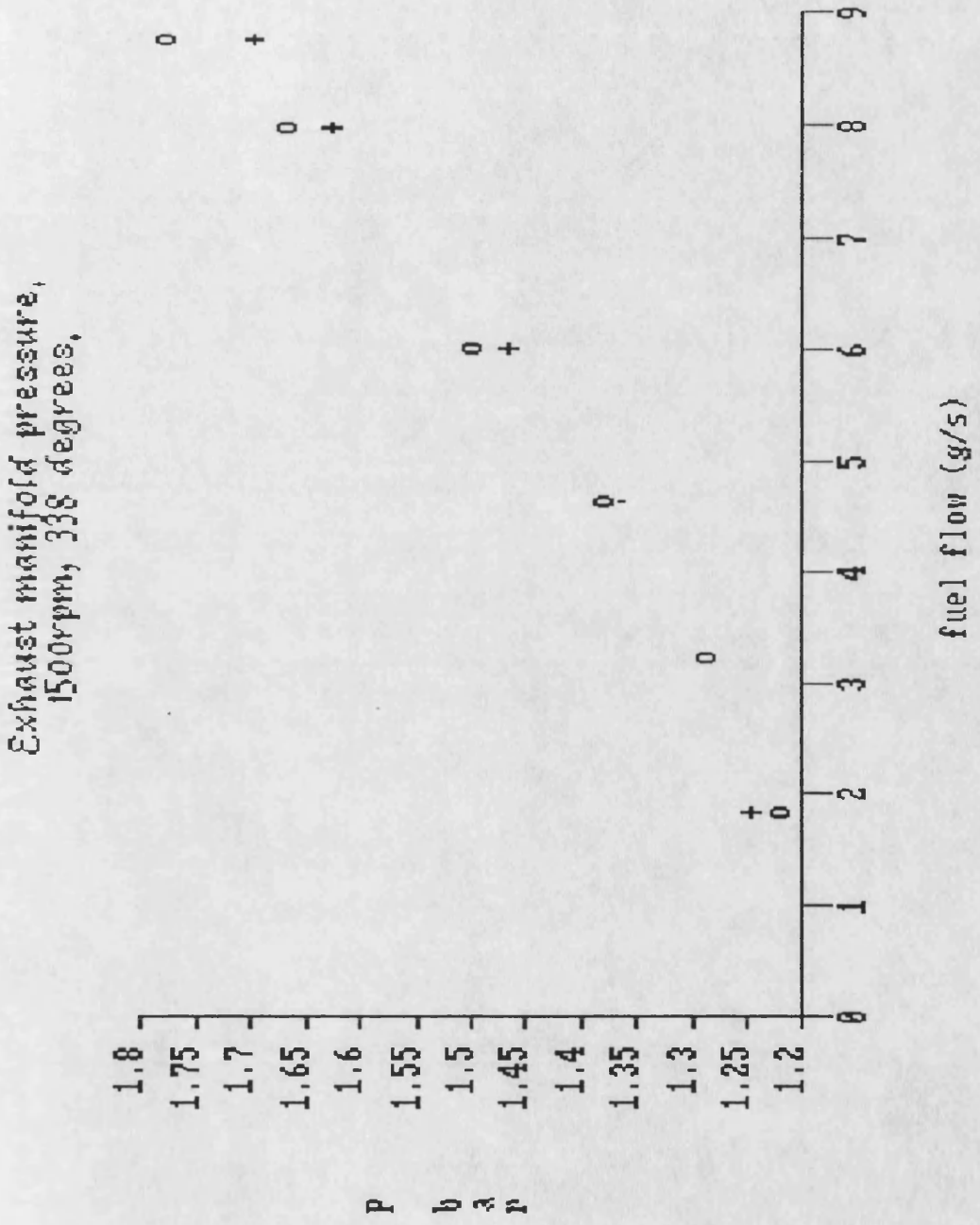


Figure 8.1j Turbine speed against fuel flow at 1500rpm for the parallel simulation (+) and the TL11 engine (o).

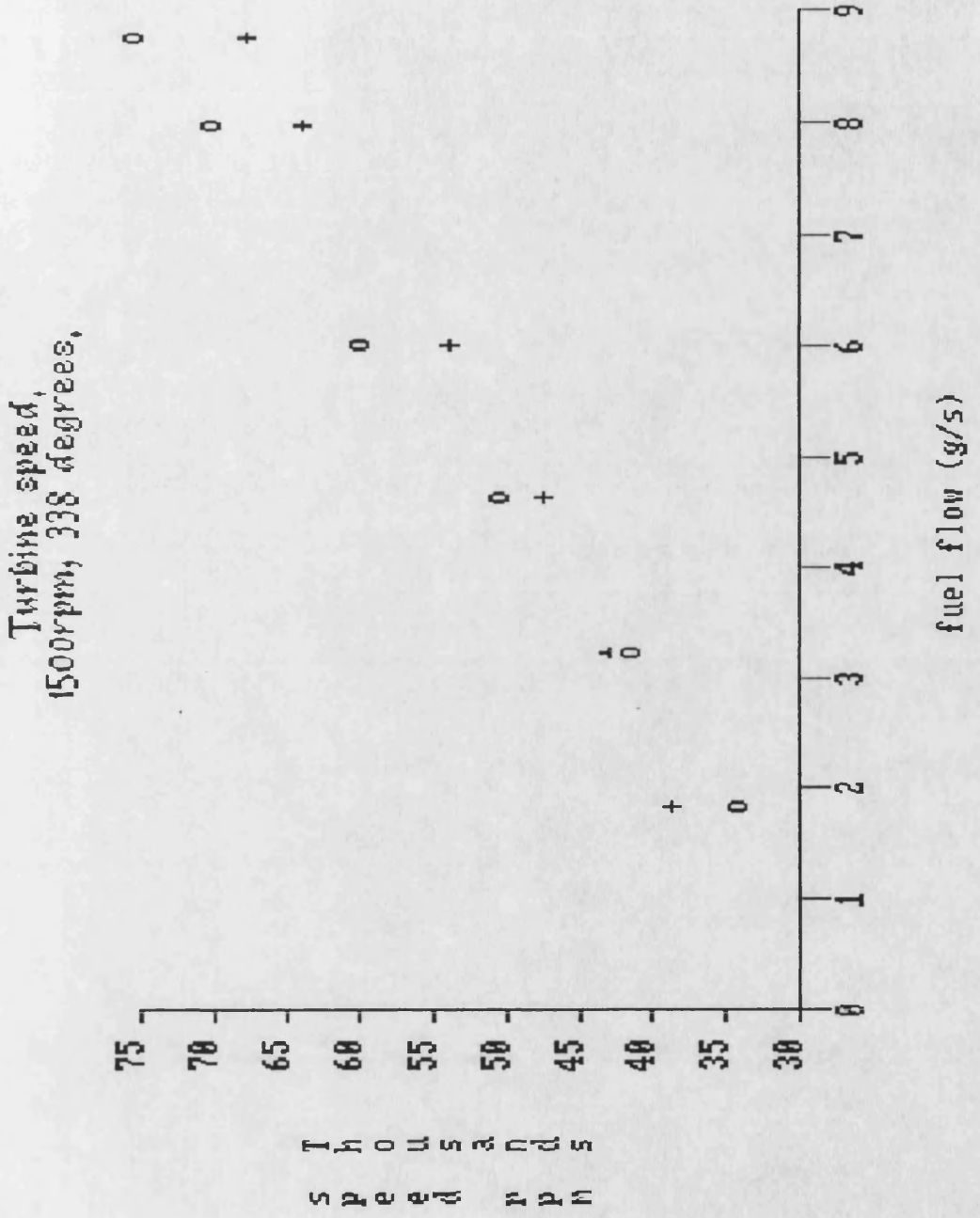


Figure 8.2a BMEP against fuel flow at 2000rpm for the parallel simulation (+) and the TL11 engine (o).

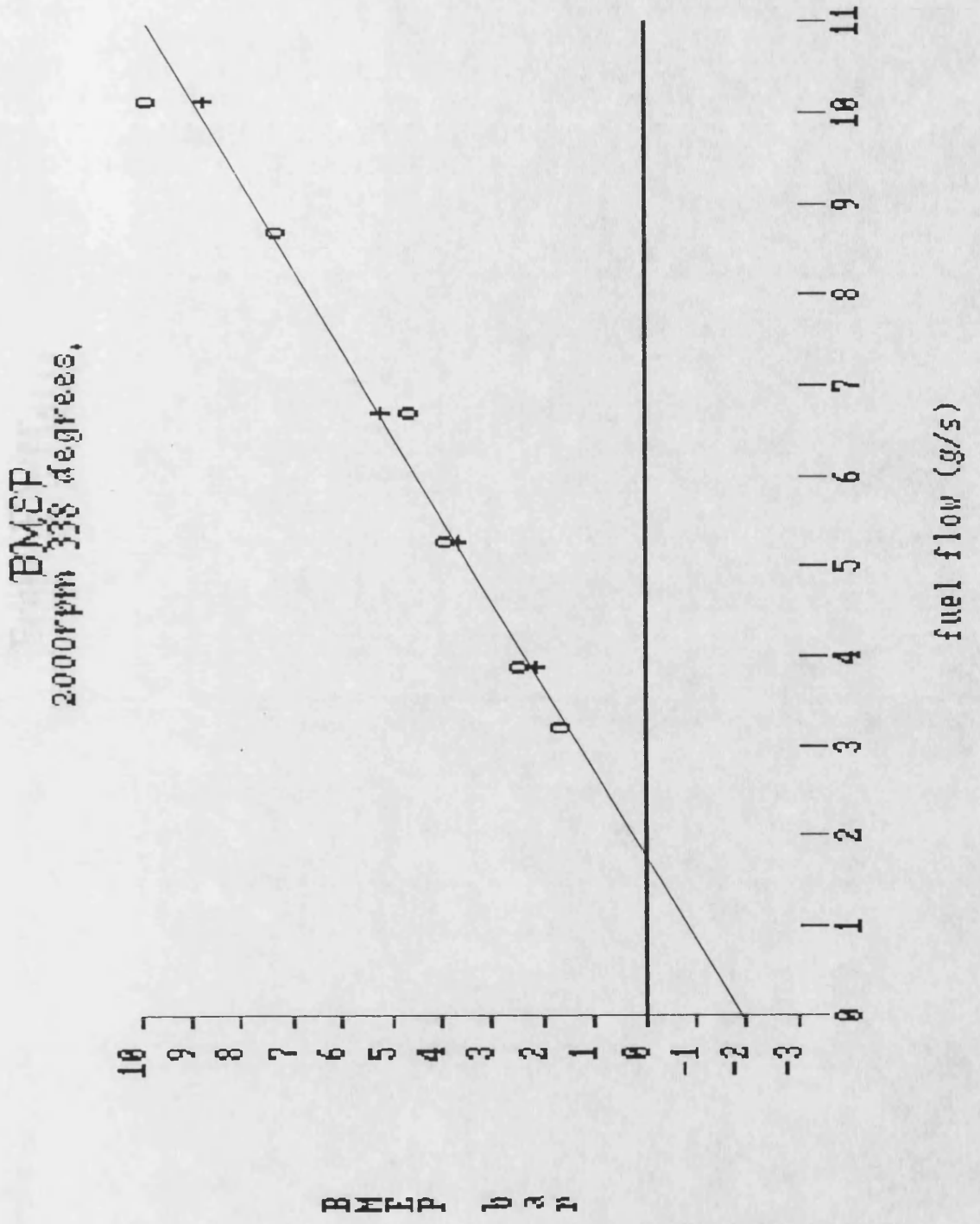


Figure 8.2b Brake Power against fuel flow at 2000rpm for the parallel simulation (+) and the TL11 engine (o).

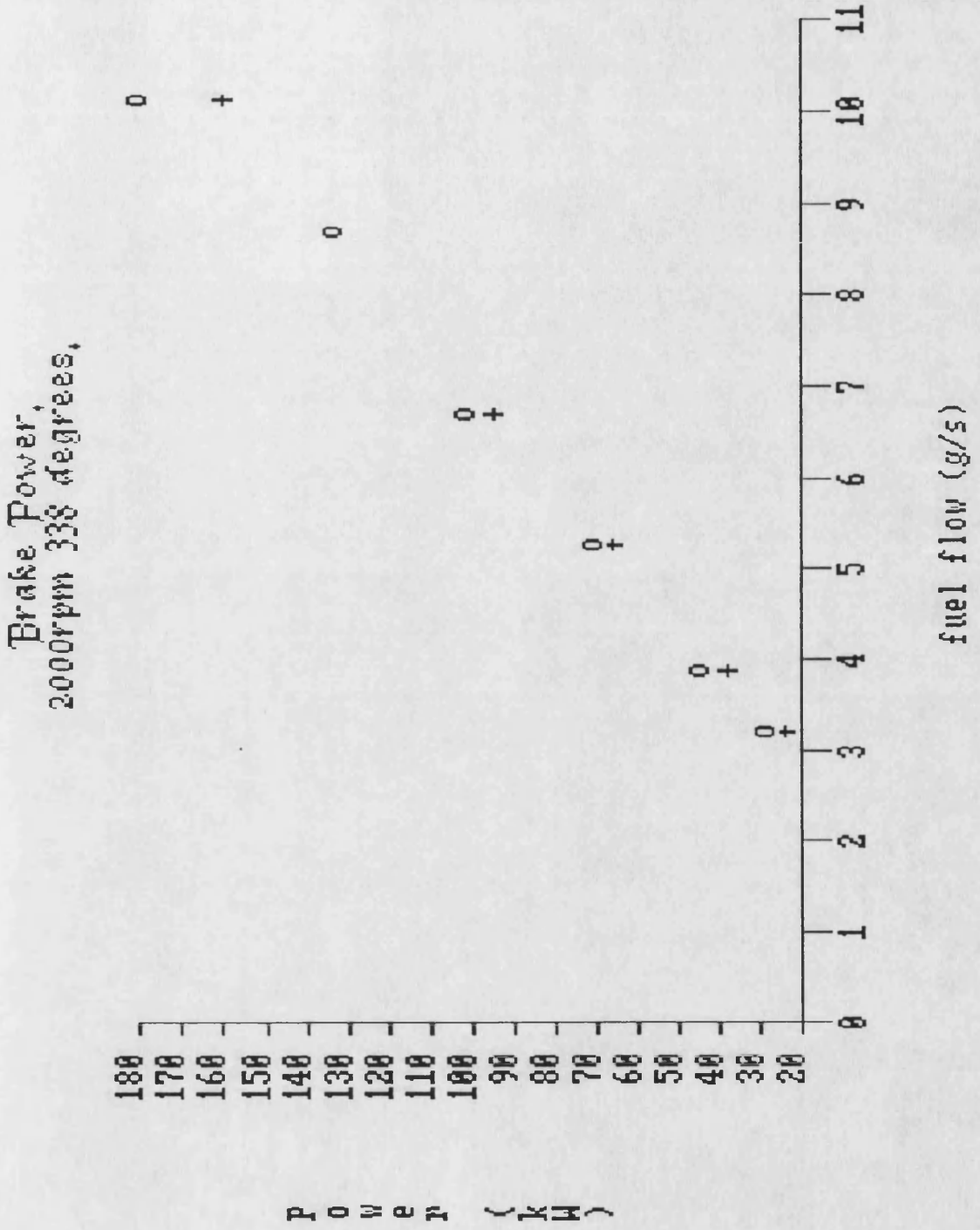


Figure 8.2c Brake efficiency against fuel flow at 2000rpm for the parallel simulation (+) and the TL11 engine (o).

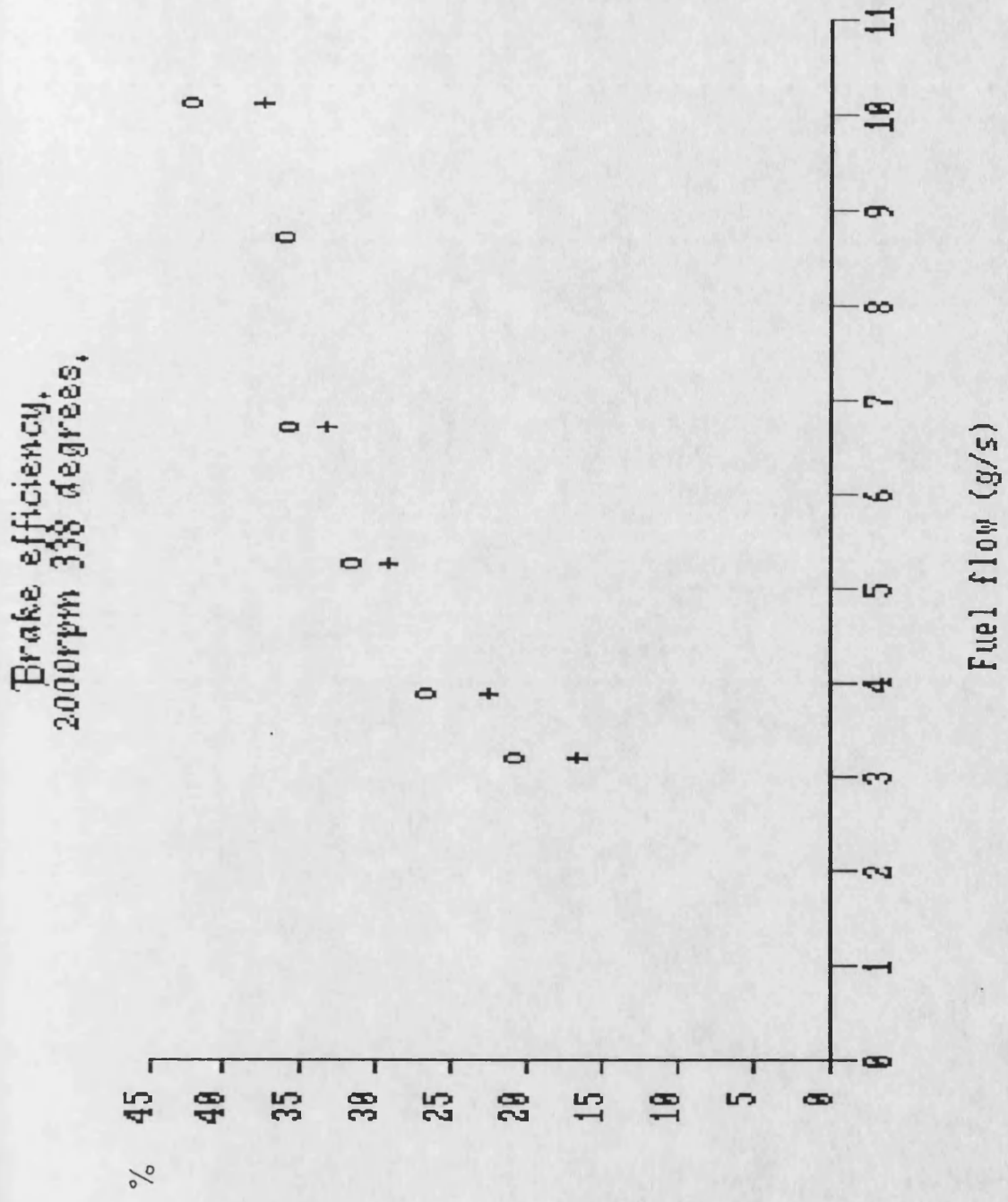


Figure 8.2d BSFC against fuel flow at 2000rpm for the parallel simulation (+) and the TL11 engine (o).

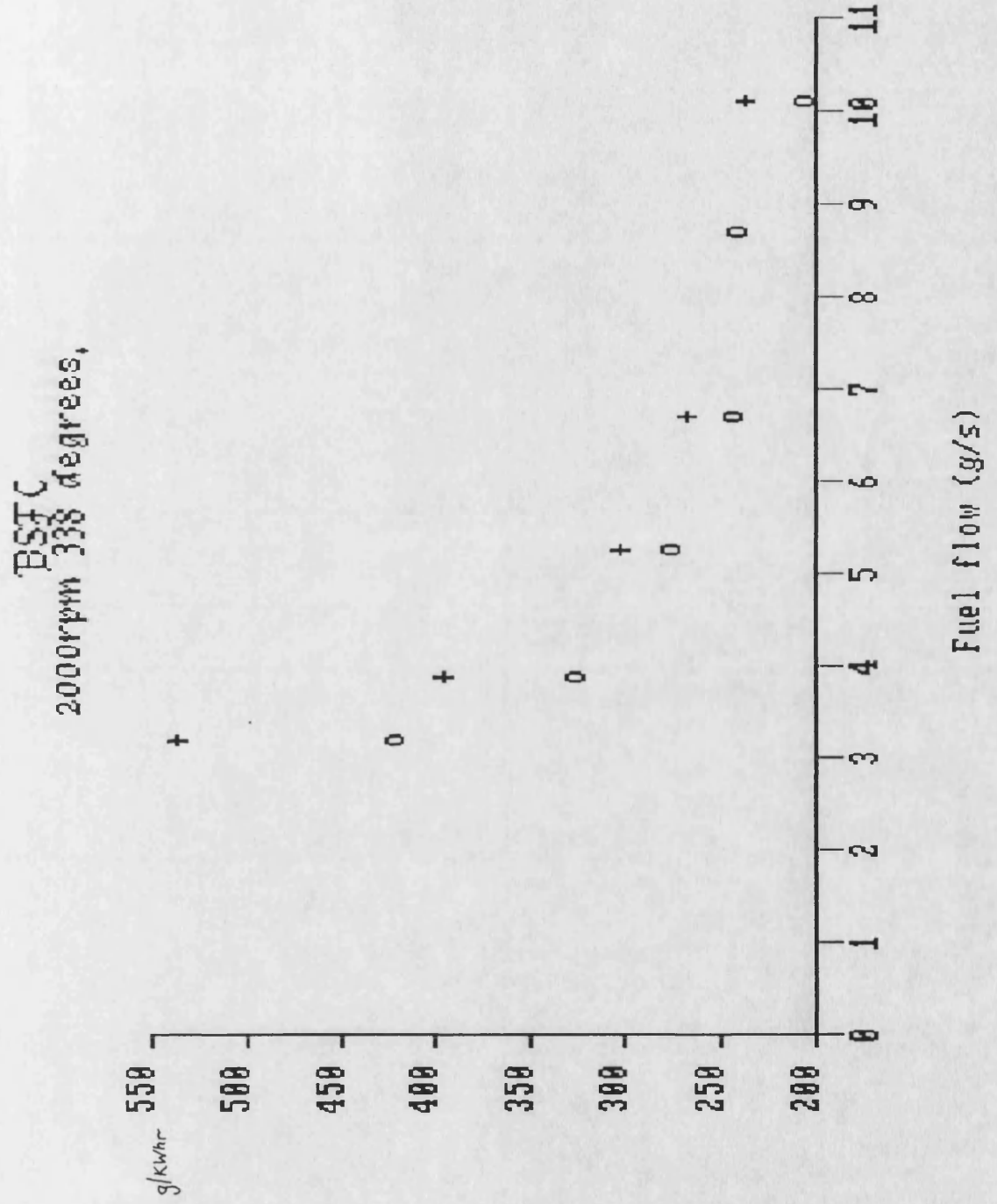


Figure 8.2e Air flow against fuel flow at 2000rpm for the parallel simulation (+) and the TL11 engine (o).

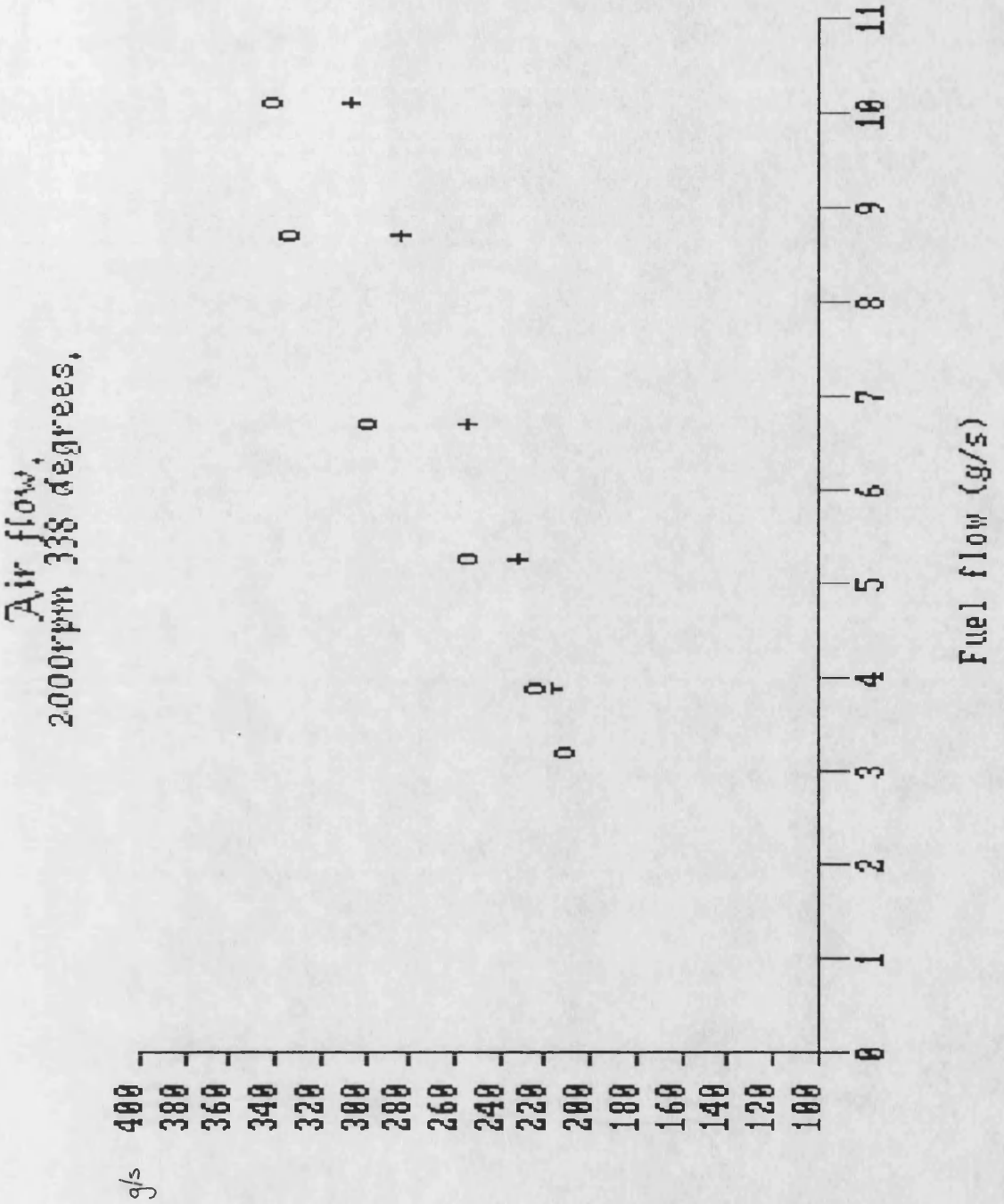


Figure 8.2f Inlet manifold temperature against fuel flow at 2000rpm for the parallel simulation (+) and the TL11 engine (o).

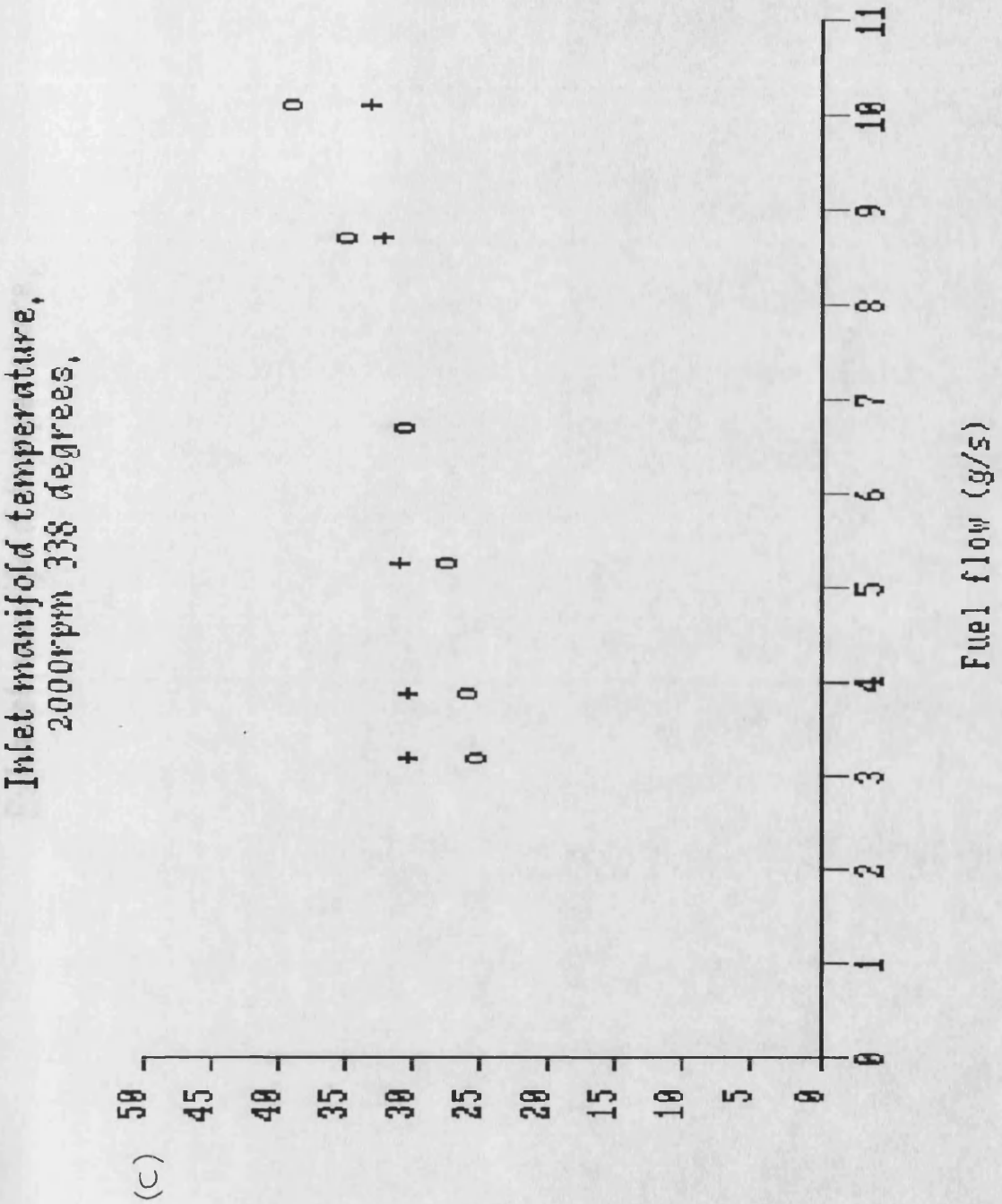


Figure 8.2g Exhaust manifold temperature against fuel flow at 2000rpm for the parallel simulation (+) and the TL11 engine (o).

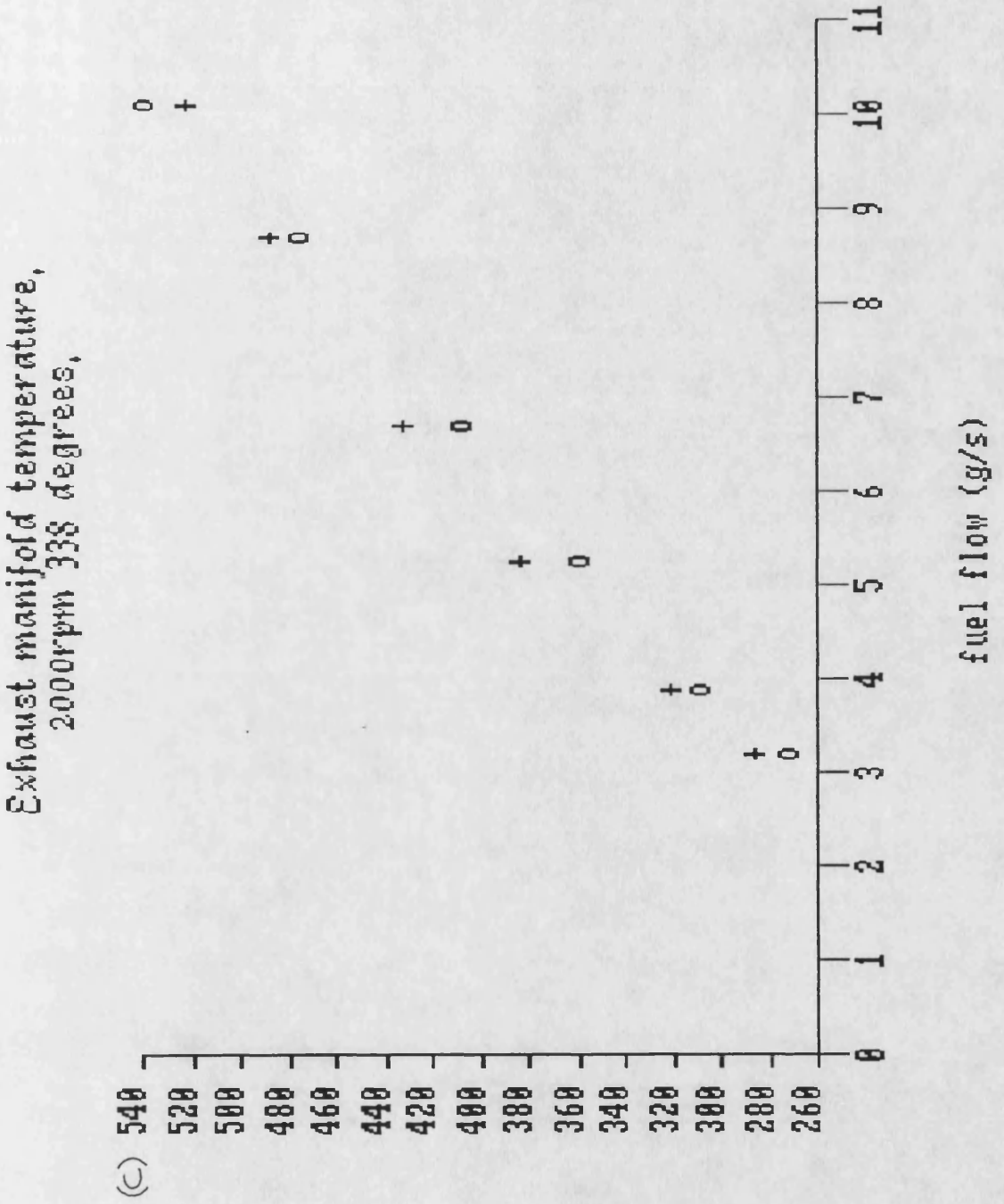


Figure 8.3 The cylinder needle lifts against crankshaft angle for the variable fuel injection experiment.

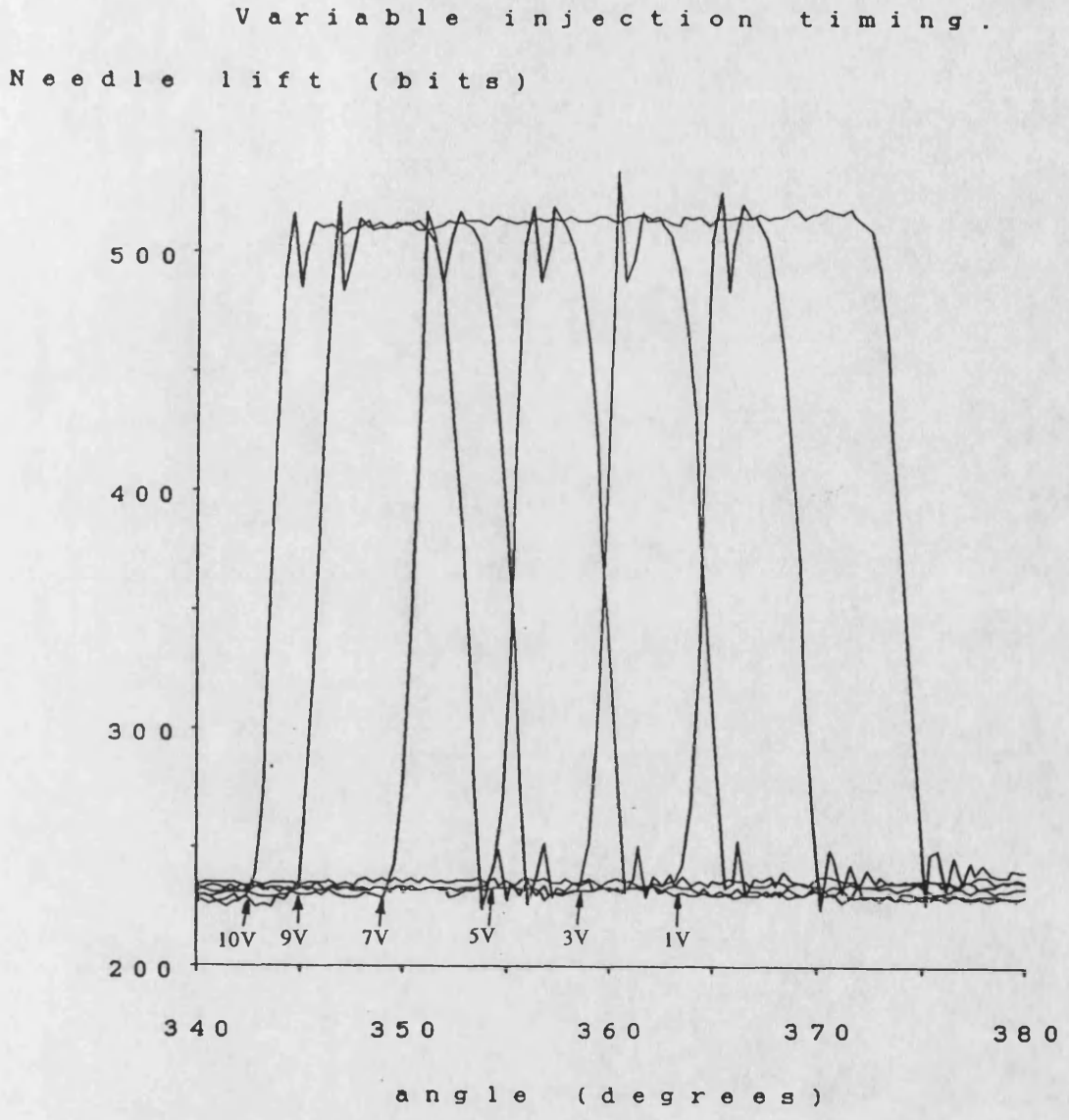


Figure 8.4a Cylinder pressure against crankshaft angle with an injection timing input of 10V. Engine speed 1500rpm. Torque = 500Nm.

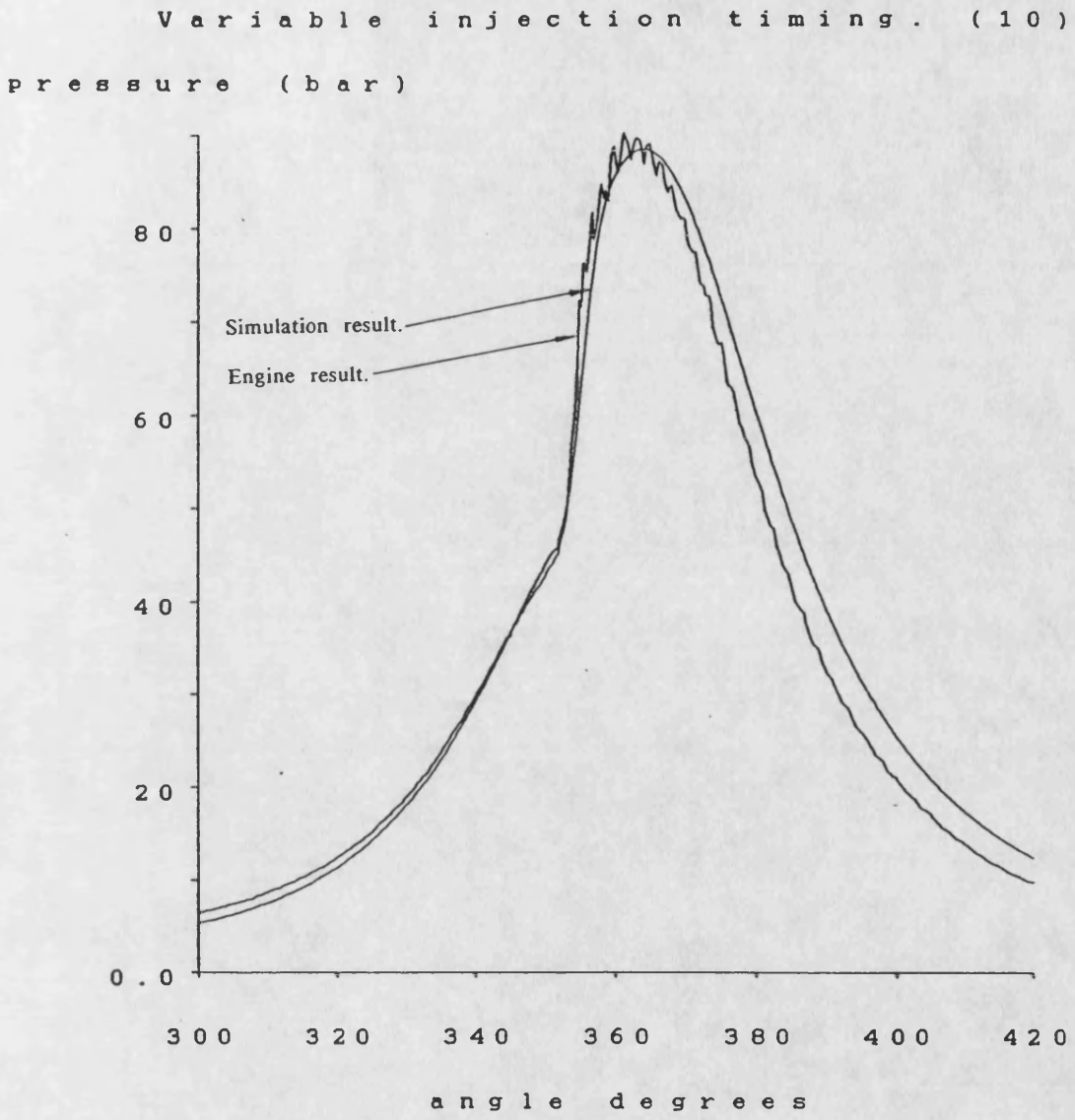


Figure 8.4b Cylinder pressure against crankshaft angle with an injection timing input of 9V. Engine speed 1500rpm. Torque = 500Nm.

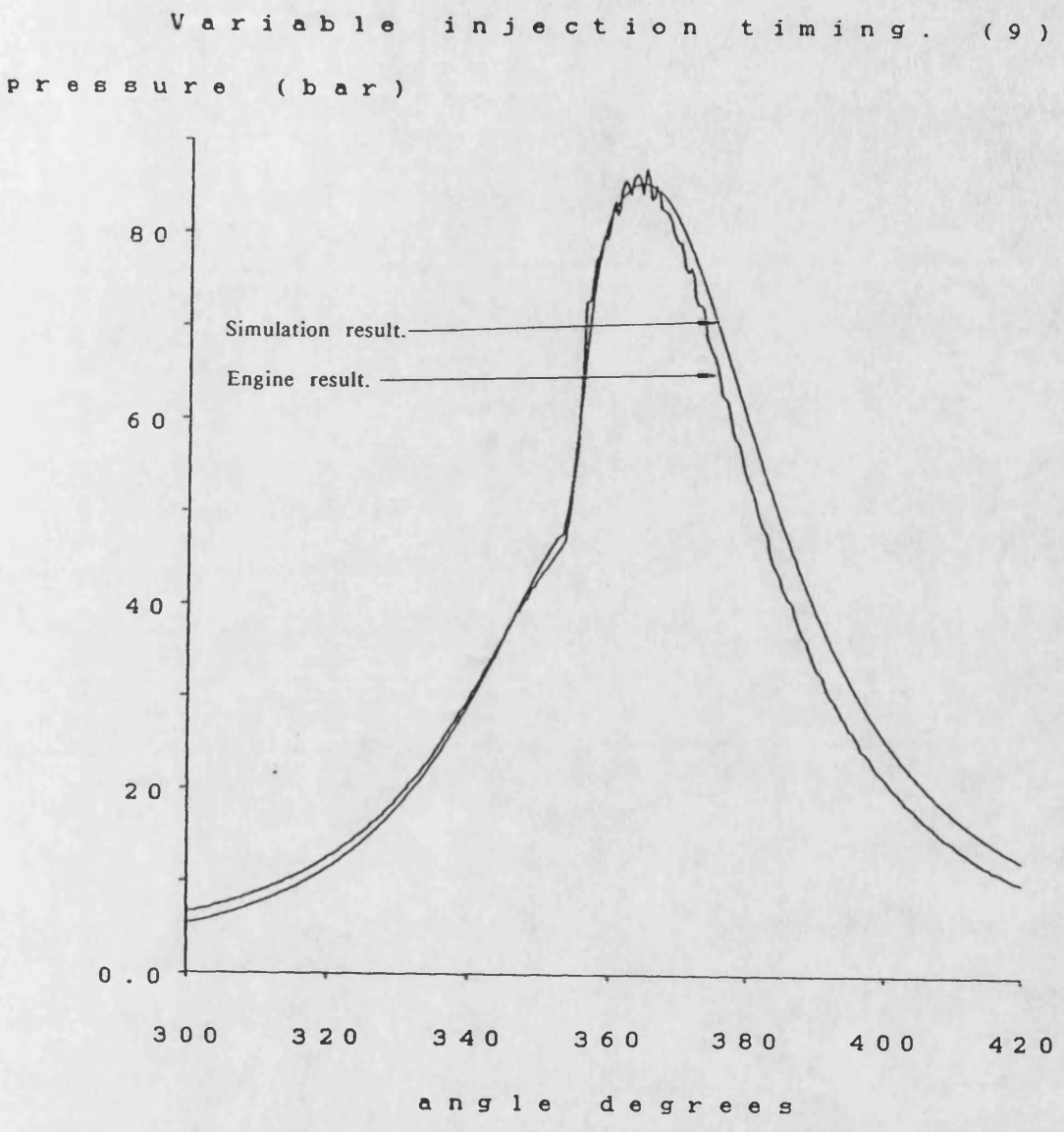


Figure 8.4c Cylinder pressure against crankshaft angle with an injection timing input of 7V. Engine speed 1500rpm. Torque = 500Nm.

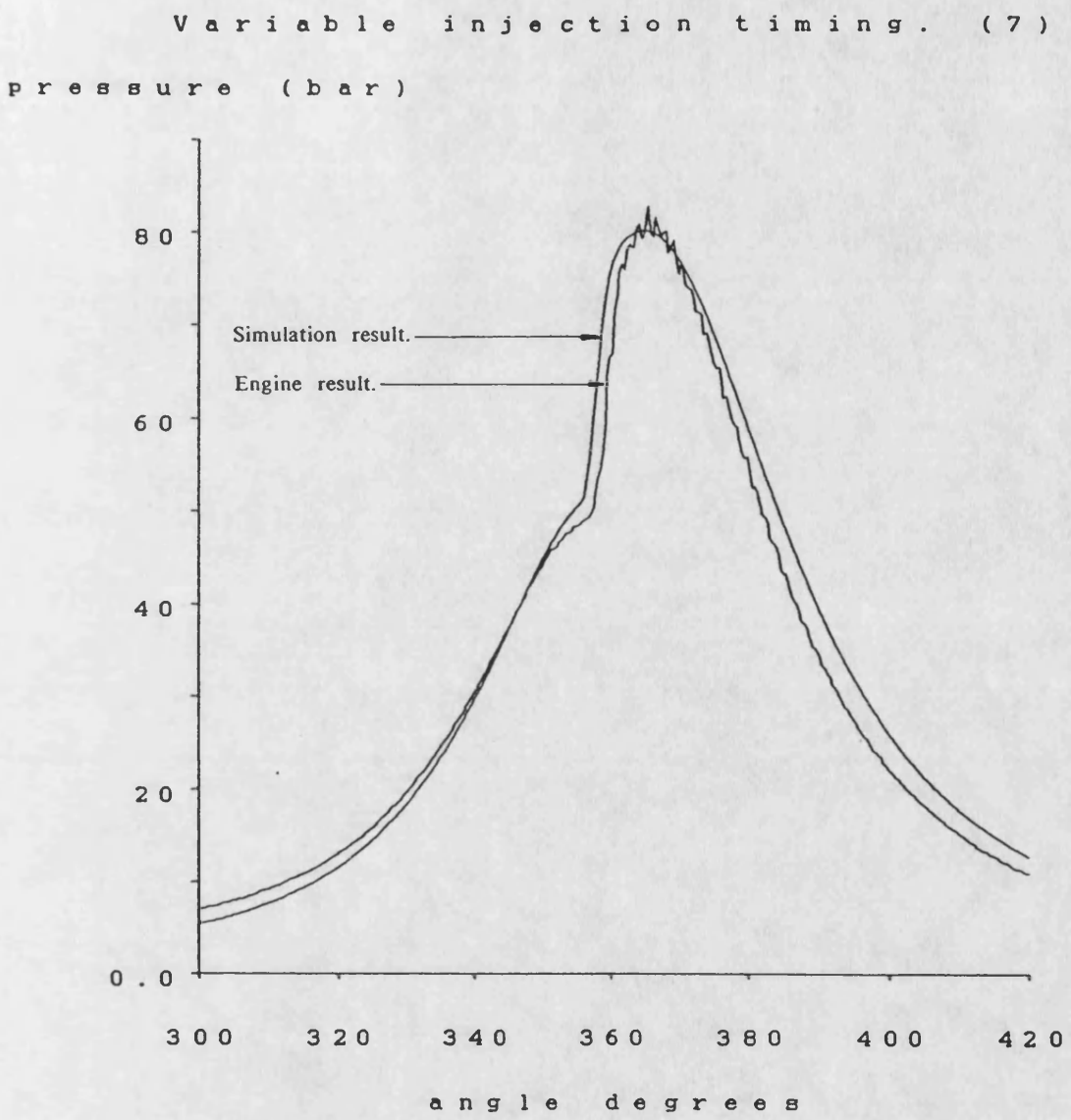


Figure 8.4d Cylinder pressure against crankshaft angle with an injection timing input of 5V. Engine speed 1500rpm. Torque = 500Nm.

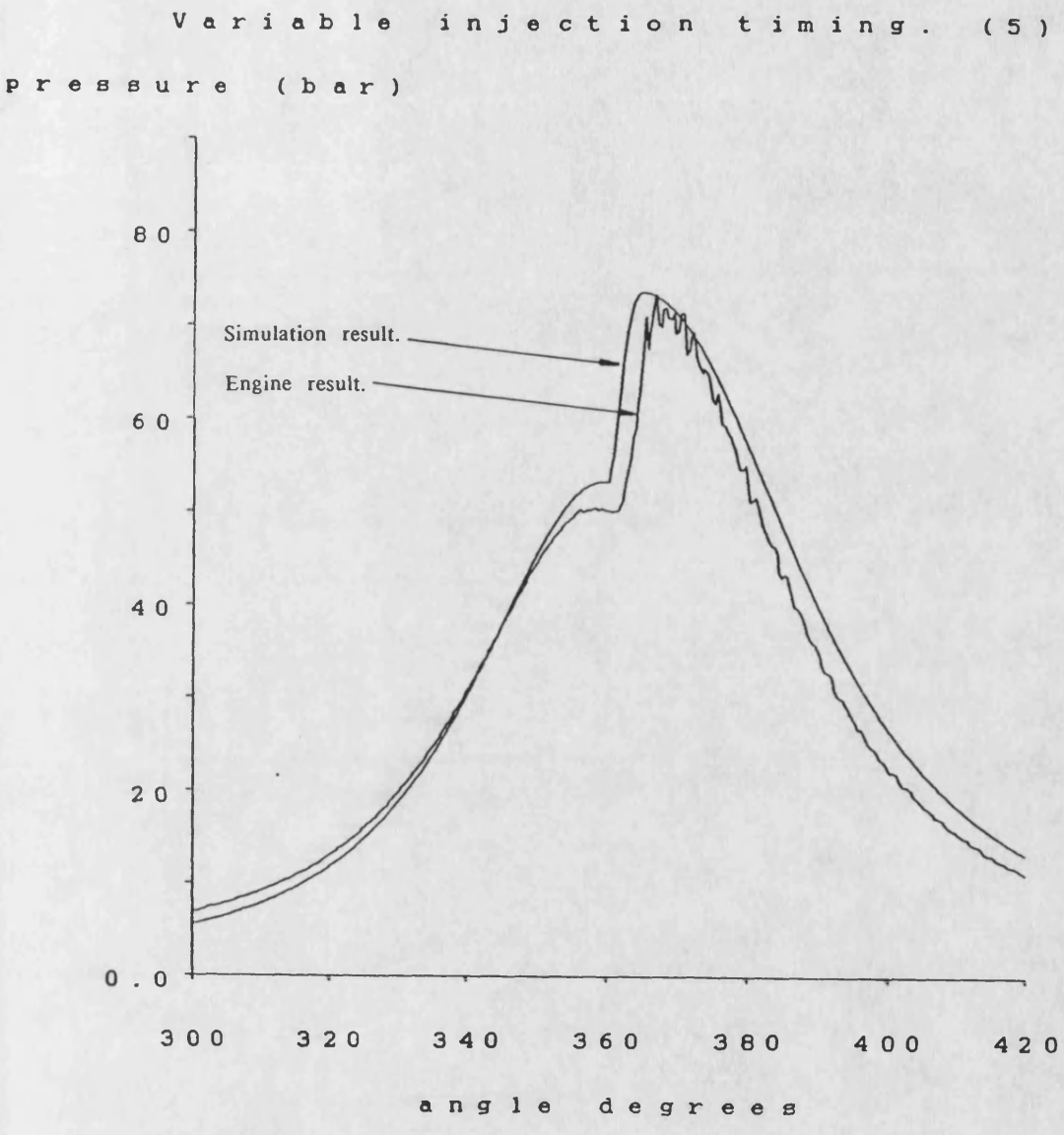


Figure 8.4e Cylinder pressure against crankshaft angle with an injection timing input of 3V. Engine speed 1500rpm. Torque = 500Nm.

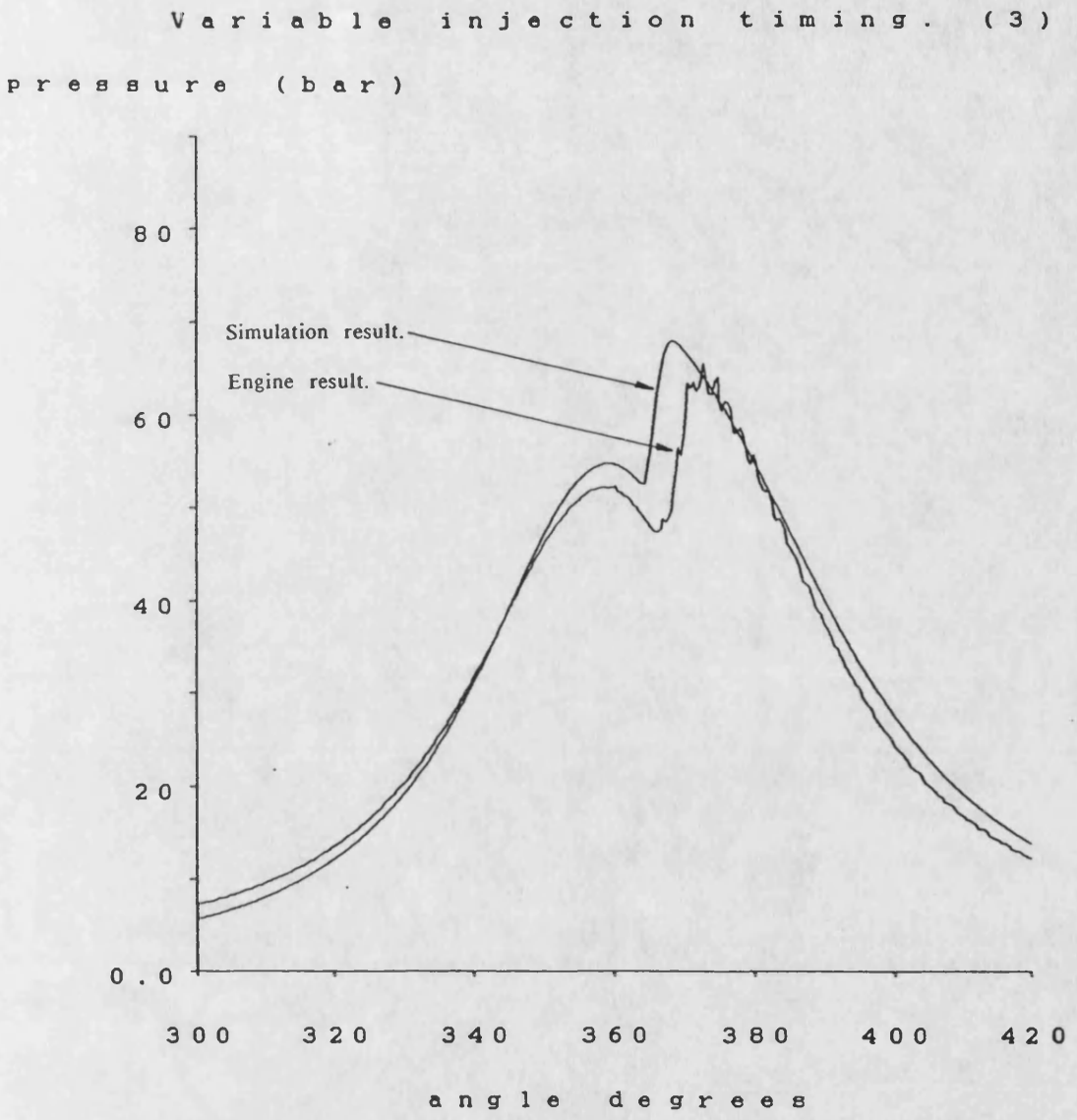
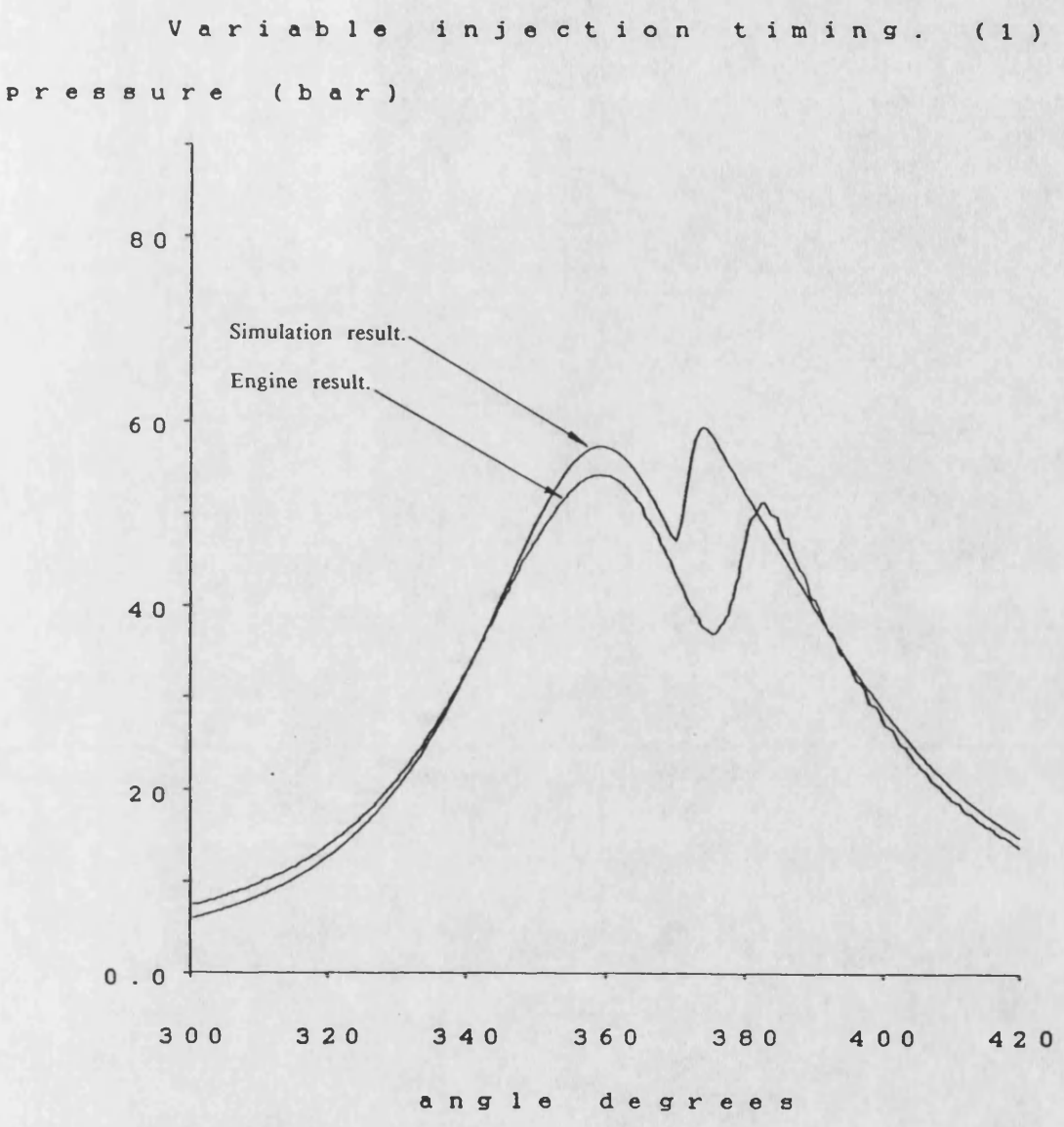


Figure 8.4f Cylinder pressure against crankshaft angle with an injection timing input of 1V. Engine speed 1500rpm. Torque = 500Nm.



CHAPTER 9.

THE APPLICATIONS OF A HIGH SPEED SIMULATION OF A DIESEL ENGINE.

9.1. Introduction.

Diesel engine simulation using complex engine models, such as the 'filling and emptying' method, has tended to be used in applications where the slowness and the 'off line' nature of the simulation can be accepted. This limits the use of complex engine models to areas such as basic engine design. The application of parallel processing to complex diesel engine simulation in this thesis has shown that considerable improvements in simulation speed can be achieved. The simulation user can now interact with the engine simulation as it runs. The development of the new parallel computing hardware in chapter 3 and the new parallel software in chapter 6 gives the possibility of an even faster 'real time' simulation. The engine simulation would progress at the same rate as the engine which is being simulated. This would further enhance the possibilities for user interaction with the simulation.

The increase in engine simulation speed that the use of parallel processing has given allows new applications for engine simulation to be identified. These are applications which were formerly precluded by the slow speed of simulation on a serial computer. Possible new applications for a fast engine simulation are discussed in the next four sections.

9.2. Engine condition monitoring and fault diagnosis in marine engines.

Modern maritime diesel engines are maintained on a 'running hours' basis where the parts of an engine are replaced after a defined lifetime. The cost of 'running hours' maintenance is high in terms of spare parts and manpower. The 'running hours' maintenance of an engine can often mean the replacement of non-faulty engine parts, and the overhaul itself may introduce new faults into the engine.

The alternative strategy of 'breakdown' maintenance, where the engine is overhauled when something goes wrong, cannot be used on a ship because of the safety implications. A third type of maintenance strategy is 'condition based' maintenance, where the overhaul of an engine is determined by the condition of the engine parts. This can extend the time between overhauls and the life of engine parts whilst maintaining the reliability of the engine. It can also allow fault trends to be identified early, leading to efficient planning of engine maintenance.

The full implementation of 'condition based' maintenance is difficult. This is due to the difficulty of finding an adequate reference with which the performance of a real diesel engine can be compared and contrasted. A complete database of diesel engine operating conditions acquired from an engine test-bed would be very large. Katsoulakos et al. [62] [63] note that a database of engine test-bed results for a large ship engine would probably be incomplete due to the huge variations in operating and environmental conditions that such an engine would meet. Engine test-bed measurements can also significantly differ from the actual performance recorded on board a ship. Uitermarkt [64] notes that these changes are due to the difference in operating conditions between a test-bed cell and those on board the ship. The changes can include differences in the installation of cooling systems and exhaust pipe lengths, differences in the types of engine load, and differences in the types of fuel and lubricating oils being used in the engine.

It is envisaged that a complex diesel engine simulation could be used as a detailed data reference for a real diesel engine. The simulation could take account of all the environmental and load conditions giving a complete picture of how the engine should be running. The detailed simulation could also be used in the diagnosis of engine faults. Faults that effect the thermodynamic and fluid dynamic processes of the engine can be simulated using a complex engine model such as the 'filling and emptying' method. These faults can therefore be identified by comparing the results from a simulated faulted engine and the real faulted engine. Possible engine faults that could be simulated are listed below:

- Air cleaner and exhaust silencer blockages.
- Leaking inlet and exhaust manifolds.
- Fouled or blocked inter-cooler.
- Sticking fuel injectors.
- Poor quality fuel being burnt.
- Inadequate engine cooling due to poor coolant flow.
- Piston ring wear and therefore piston blow-by.
- High engine and turbocharger friction.
- Leaking cylinder inlet and exhaust valves.
- Incorrect calibration of fuel injection equipment.

The method of fault simulation will vary depending on the fault. For example, leaks in the manifolds and the cylinders can be simulated as variable area orifices through which gases can flow. Blockages can be modelled as restrictions to gas flow between engine simulation control volumes. It is envisaged that these faults would be permanently built into the simulation software. Values for the fault parameters would be set to give no effect for normal engine operation.

The possible number of engine faults is large and the characteristics of these faults can vary considerably depending on the operating condition of the engine and the size of the engine fault. The diagnosis of an engine fault will therefore require the simulation of a large number of fault conditions with varying sizes of fault. This will take a great deal of time using an engine simulation on a conventional serial computer. Using the engine simulation on a parallel computer, the time needed for fault identification would be greatly reduced thus improving the effectiveness of the technique. O'Leary [65] stresses the importance of any health monitoring system operating in 'real time' to the speed of the engine degradation processes.

It is envisaged that a condition monitoring system based on a reference diesel engine model would be used in conjunction with an expert system. The expert system would be responsible for the interpretation of the results from the diesel engine. It would use the engine model during normal operation for the optimisation of engine operating parameters. When the results from the engine deviated from normal the expert system would also be responsible for the analysis of the fault and the selection of appropriate fault models for the diagnosis of the engine fault. The order in which probable faults are selected by the expert system will be important. Sensible choice of probable engine faults will minimise the number of fault conditions to simulate and thus the time needed for fault identification. The application of expert systems to condition monitoring using a detailed engine model is described by Katsoulakos et al. [62] [63]. A schematic diagram of a possible engine condition monitoring system is given in figure 9.1.

Simple engine condition monitoring using engine test-bed results has already been used commercially. The company MaK [66] [67] have implemented an engine condition monitoring scheme based on a simple engine model. This model is based on engine test-bed results and makes adjustments to these to take account of operating conditions and environmental factors. BP shipping [68] have also implemented a condition monitoring system for diesel engines on board some of their ships. In both cases the condition monitoring is based on indicated mean pressure for the engine cylinders. BP shipping achieved savings of £19,250 per annum for an investment of £30,000 using this form of condition monitoring. Substantial commercial benefits are therefore already possible through the use of engine condition monitoring.

9.3. A high performance engine simulation as an interactive engine design aid.

Engine simulation using complex engine models is slow and non-interactive. This slowness can limit the use of engine simulation as a general design tool. With a high speed engine simulation the diesel engine design engineer can be given interactive control of his designed engine. This can encourage the engineer to experiment and try novel designs. The 'off line' nature of the present engine simulation precludes this experimentation.

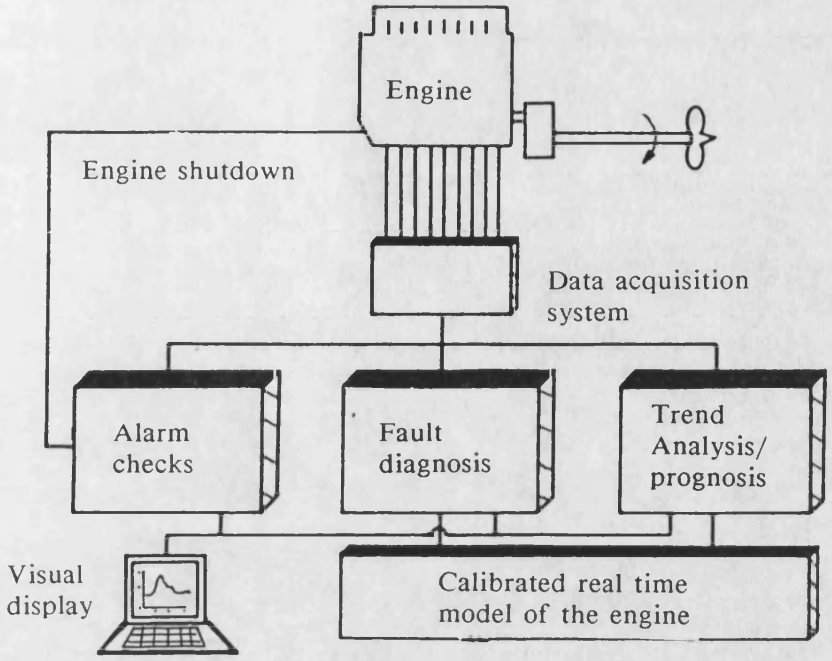
9.4. High performance engine simulation in advanced engine control design.

The design of new engine management systems and the instrumentation of existing engines for advanced control is costly. However the 'off line' nature of present complex engine simulations make its use as a design aid rather limited since the designer would prefer a degree of interaction with the model. A fast simulation would allow the quick testing of advanced control strategies for the diesel engine allowing interaction and exploration of the problem. Interesting control strategies could be refined prior to full implementation on a real engine.

9.5. Engine operator training.

A detailed 'real time' simulation would allow in depth training for diesel engine operating personnel. Engine operators would be able to be trained to handle unforseen circumstances. Trainees would see the effect of their actions without endangering a real engine and be able to learn from their mistakes. Similar work has been carried out by Dale [23] and Berry [24] on electrical power grid simulation using the Bath University Parallel Computer. This is now to be used for grid control training for the National Grid Company.

Figure 9.1 A possible engine condition monitoring system.



CHAPTER 10.

CONCLUSIONS.

10.1. Conclusions.

A high speed simulation of a Diesel engine using a parallel computer has been produced for this thesis. The new simulation has a structure in which control is distributed across all of the processors in the computer. Previous parallel simulations used a centralised structure in which one processor controlled all of the other processors. The new simulation allows a high processor utilisation and very efficient communication between the parallel sections of the simulation. Because of its distributed structure the new simulation is more exact than the previous parallel simulation developed by Jones at the University of Bath.

The new parallel computer engine simulation has been compared to an existing serial computer 'filling and emptying' engine simulation. The results from both simulations were in agreement. The simulation has also been compared with results from a real engine. A good correlation has been shown between the engine and the simulation at a variety of load, speed and injection timing conditions. This correlation was made with no effort to specially tune the various empirical parameters in the simulation.

For the purpose of validation of the parallel simulation with an engine, a new data acquisition system has been designed. This has been successfully used for the acquisition of results from the engine.

A new graphical display has been developed to cope with the increased speed of the new simulation. A menu based user interface has also been designed for the display of engine cycle data and the control of the simulation.

10.2. Future work.

The new parallel simulation has highlighted a number of limitations in the present hardware design based on the MC68020 processor. A new computer based on the Transputer has been developed at the University of Bath. The new simulation described in this thesis should now be transferred to the new computer.

The new parallel simulation is limited to simulation of the Leyland TL11 engine. Serial computer simulations, such as SPICE, allow the user to easily modify the type of engine modelled. The development of an adaptable front end for the parallel simulation will be very important to its future use.

BCPL has been used to write the parallel simulation. This is a very flexible programming language but is not supported commercially. The transfer of the parallel simulation into ANSI 'C', which is required for the transfer to the new Transputer computer, should allow the wider use of the simulation.

The new parallel simulation has been compared to the Leyland TL11 engine for a few engine operating conditions. A more complete analysis will need to be carried out. This should include a full mapping of the engine across its speed load range and a comparison with the equivalent map for the parallel simulation.

A number of empirical models are used in the new parallel simulation. These empirical models have constants which can be changed to correlate with specific engines. The constants used at present are those suggested by the authors of the models. From a thorough analysis of results from the TL11 engine more appropriate constants and possibly better empirical models could be found.

Chapter 6 described a number of new parallel integration methods for Diesel engine simulation. With the increased number of processing nodes available in the new Transputer computer these methods should be applied to the solution of a multi-cylinder diesel engine simulation.

The thesis has discussed the possible use of a high speed simulation for engine condition monitoring. A natural extension of this work will be to use expert systems for the identification of engine operating conditions and faults in conjunction with the simulation.

REFERENCES.

1. EATON, J. SMITHERS, J. CURRAN, S.
This is IT, A manager's guide to information technology.
Phillip Allan Publishers Ltd, Oxford. 1988.

2. SMITHS ASSOCIATES,
International conference on parallel processing.
The Royal Society, London. 9th/10th December 1987.

3. STONE, R.
Introduction to internal combustion engines.
Macmillan publishers Ltd. 1985.

4. LUCAS, G G. EMTAGE, A L.
Microprocessor control of the hydrogen/petrol engine.
Proceedings of the International conference,
Computers in Engine Technology.
I.Mech.E. March 1987.

5. WALLACE, F J.
Variable geometry turbocharging control under transient conditions.
SAE 810336. 1981.

6. MACCARLEY, C A. MEYER, D G.
An auxiliary sensor processor to provide real time fuel delivery feedback for a microprocessor based diesel engine controller.
SAE 870435

7. CLARK, C A.
Electronics applied to systems using internal combustion engines; from lawnmowers to low speed marine engines.
I.Mech.E. seminar: The benefits of electronic control systems for internal combustion engines.
January 1989.

8. JONES, A D.
The application of parallel processing to diesel engine modelling.
Ph.D. Thesis University of Bath. 1987.

9. WATSON, N. & JANOTA, M S.
Turbocharging the internal combustion engine.
Macmillan Press, London. 1982.

10. GARNISH, B R.
The estimation of discharge coefficients for poppet valves.
English Electric Company, Whetstone. December 1967.

11. HOHENBURG, G F.
Advanced approached to heat transfer calculations.
S.A.E. Paper No. 790825. September 1979.

12. WOLFER, H.
Ignition lag in diesel engines.
VDI Forschungsheft No. 392 1938.
Translation by Royal Aircraft Establishment, Farnborough.
Library No. 358. August 1959, UDC No. 621-436 047.

13. WATSON, N.
E17 Combustion and Gas properties.
Department of Mechanical Engineering, Imperial College, London. September 1979.

14. WATSON, N. PILLEY, A D. & MARZOUK, M.
A combustion correlation for diesel engine simulation.
S.A.E. Paper No. 800029, 1980.

15. NEWELL, H K. STARKMAN, E S.
Thermodynamic properties of octane and air fo engine performance calculations.
SAE Progress in Technology series Vol. 7. 1964.

16. CHARLTON, S J.
SPICE User Manual. University of Bath, April 1986.

17. KRIEGER. R B. BORMAN, G L.
The computation of apparent heat release for internal combustion engines.
ASME 66-WA/DGP-4 1966.

18. ROBERTS, E W.
Variable geometry turbochargine optimisation and control.
Ph.D. thesis, University of Bath. 1984.

19. MILLINGTON, B W. & HARTLES, E R.
Frictional losses in diesel engines.
S.A.E. Paper No. 680590. 1968.

20. CHEN, S K. FLYNN, P.
Development of a compression ignition research engine.
S.A.E. 650733. 1965.

21. BENSON, R S. HORLOCK, WINTERBONE, D E.
The thermodynamics and gas dynamics of IC engines.
Oxford University Press, 1982.

22. MEGUERDICHIAN, M. & WATSON, N.
Prediction of mixture formation and heat release in diesel engines.
S.A.E. Technical paper 780225. February 1978.

23. DALE, L A.
Real time modelling of multi-machine power systems.
Ph.D. thesis, University of Bath. 1986.

24. BERRY, T.
Real time modelling of complex power systems using parallel processing.
Ph.D. thesis, University of Bath. 1989.

25. MOTOROLA.
MC68000 8-/16-/32- Bit microprocessors.
Prentice Hall Inc, 1986.

26. HITACHI
HD68450 Direct memory access controller. Specification.
Hitachi electronic components.

27. MOTOROLA.
MC68020 32 Bit microprocessor. User's manual.
Prentice Hall Inc, 1985.

28. MOTOROLA.
MC68881. Floating point coprocessor. User's manual.
Prentice Hall Inc, 1985.

29. COONEN, J L.
A proposed standard for binary floating point arithmetic.
ACM Signum Newsletter. October 1979.

30. HOPGOOD, S R A. DUCE, D A, GALLOP J R, SUTCLIFF, D C.
Introduction to graphics kernel systems (GKS).
Academic Press, 1983.

31. Multilink user's guide.
Ninetiles computer systems Ltd. 1984

32. RICHARDS, M.
Tripos - A portable operating system for minicomputers.
Software Practice Experience, Vol. 9, February 1979.

33. KING, T J.
TRIPOS User guide. Issue 2.
School of Mathematics, University of Bath. 1983.

34. KING, T J.
TRIPOS Programming guide. Issue 2.
School of Mathematics, University of Bath. 1983.

35. KING, T J.
TRIPOS technical manual. Issue 2.
School of Mathematics, University of Bath. 1983.

36. RICHARDS, M. & Whitby-Stevens, C.
BCPL - The language and its Compiler.
Cambridge University Press, 1980.

37. EMERY, G.
BCPL & C.
Blackwell Scientific Publications, Oxford. 1986.

38. DUNN, R W. DANIELS, A R. GOTT, V S. SELWYN, C G.
A new architecture of high performance parallel computer for use in condition monitoring of large diesel engines.
Proceedings of the 2nd international conference on software engineering for real time systems.
I.E.E. Conference publication No. 309. September 1989.

39. HAYSOM, F J. DUNN, R W. DANIELS, A R. CHARLTON, S J.
Real time simulation in the health monitoring of diesel engine power plant using parallel processing.
I.Mech.E. seminar: The benefits of electronic control systems for internal combustion engines.
January 1989.

40. DANIELS, A R. DUNN, R W. GOTT, V S. & SELWYN, C G.
Real time simulation of diesel engines using the T800 transputer.
SERC/DTI Transputer initiative workshop on transputer development environments.
Abingdon, Oxon. Nov/Dec 1987.

41. PERIHELION SOFTWARE LTD,
The HELIOS operating system.
Prentice Hall International(UK). 1989

42. HARBISON, S P. & STEELE, G L.
C: A reference manual. 2nd edition.
Prentice Hall Inc. 1987.

43. ANNAND, W J D.
Choice of a computing procedure for digital computer synthesis of reciprocating engine cycles.
Journal of Mechanical engineering science. Vol. 10, No. 3. 1968.

44. HALL, G. WATT, J M. 1976.
Modern numerical methods for ordinary differential equations.
Chap. 1. pp 2-19
Oxford University Press.

45. SELWYN, C G.
Personal communication. 1989.

46. MIRANKER, W L. & LINIGER, W.
Parallel methods for the numerical integration of ordinary differential equations.
Mathematical Computing. Vol. 21. 1967.

47. KERCKHOFFS, E J H.
Parallel algorithms for ordinary differential equations: An introductory review.
Proceedings of the 1986 summer computer simulation conference, Reno NV USA July 1986.

48. SHAMPINE, L F. & WATTS, H A.
Block implicit one step methods.
Math. Comp., 23. pp731-740. 1969

49. ROSSER, J B.
A Runge-Kutta for all seasons.
SIAM Review, Vol. 9, No. 3, July 1967.

50. BIRTA, L G. ABOU-RABIA, O.
Parallel block predictor-corrector methods for ode's.
I.E.E.E transactions on computers. Vol C-36 No. 3
March 1987. pp 299-311

51. WORLAND, P B.
Parallel methods for the numerical solution of ordinary differential equations.
I.E.E.E. transactions on computers.
October 1976. pp 1045-1048

52. HALIN, H J. BUHRER, R. HALG, W. BENZ, H. BRON, B. BRUNDIERS, H-J. ISACSON, A. & TADIAN, M.
The ETH multiprocessor project: parallel simulation of continuous systems.
Simulation, Vol. 35, 1980, pp 109-124.

53. FRANKLIN, M A.
Parallel solution of ordinary differential equations.
IEEE transactions on computers, Vol C-27, May 1978,
pp 413-420.

54. DIJKSTRA, E W.
Cooperating sequential processes.
In Programming languages. (ed. F. Genuys)
Academic Press, New York. 1968.

55. BIRTA, L G. ABOU-RABIA, O.
Some variations on the BPC parallel integration method.
Proceedings of the Summer computer simulation conference,
Montreal, Canada. July 1987 pp 37-42

56. Leyland TL11 instrumentation manuals. Volumes 1,2 and 3.
Wolfson laboratory, School of Mechanical engineering, University of Bath.

57. MARZOUK, M. WATSON, N.
Some problems in diesel engine research with special reference to computer control and data acquisition.
Proc. I.Mech.E. Vol. 190 23/76 1976

58. ANALOG DEVICES.
AD574 data sheet.

59. NATIONAL SEMICONDUCTOR.
LF358 sample and hold amplifier data sheet.

60. SCAIFE, M.
Personal communication. 1989.

61. WHITE, C L.
Personal communication. 1989.

62. KATSOULAKOS, P S. NEWLAND, J. STANSFIELD, J T. & RUXTON, T.
Monitoring, databases and expert systems in the development of engine fault diagnaostics.
Journal of non destructive testing. Vol. 30, No. 4, pp 263-273.

63. KATSOULAKOS, P S. HORNSBY, C P W. & ZANCONATO, R.
DEEDS: The diesel engine expert diagnosis system.
Proceedings of the conference on Maritime communications and control.
Institute of Marine engineers. 26-28 October 1988.

64. UITERMARKT, R W P.
Engine fault diagnosis.
Marine Engineering Review, July 1984

65. O'LEARY, J P.
New opputunities in Diesel condition monitoring.
Institution of Diesel and gas turbine engineers.
Publication 445. January 1988.

66. Engine history at Dicare's heart.
Anon. The motor ship, July 1987. p26.

67. Higher efficiency through better monitoring,
Anon. The motor ship, July 1987. p25.

68. HIND, M.
Advanced maintenance on ships.
I.Mech.E. seminar: Condition based maintenance of engines.
March 1988.

69. Computerised Fault Finding,
Marine Engineering Review, June 1985.

70. HALLAM, A J. & COTTAM, S.
Computer program to predict the gas exchange process of a diesel engine.
Computer aided design. Vol. 7, No. 2, April 1975.

71. HALIN, H J.
Integration across discontinuities in ordinary differential equations using power series.
Simulation, Vol. , February 1979, pp 33-45.

72. HORIGUCHI, S. KAWAZOE, Y. & NARA, N.
A parallel algorithm for the integration of ordinary differential equations.
Proc. 1984 IEEE Conference on parallel processing.

73. JONES, A D. CHARLTON, S J. DANIELS, A R. & HAYSOM, F J.
A multi-processor diesel engine simulator for advanced diagnaostics.
SAE Paper 871696, Milwaukee, September 1987.

74. MILNE, R.
Artificial intelligence applied to condition health monitoring.
Chartered Mechanical Engineering, May 1986.

75. MIRANKER, W L.
A survey of parallelism in numerical analysis.
SIAM Review. Vol 13, No. 4, October 1971.

76. MOLCHANOV, I N. YAKOVLEV, M F. & GEETS, E G.
Some methods of integrating ordinary differential equations on a multiprocessor computer.
Cybernetics USA, Vol. 19, No. 5, Sept-Oct. 1983. pp 599-604.

77. NIEVERGELT, J.
Parallel methods for integrating ordinary differential equations.
Communications of the ACM, Vol. 7, No. 12, December 1964, pp 731-733.

78. PIMEMTEL, J R.
Real time simulation using multiple microcomputers.
Simulation. March 1983. pp 93-104.

79. WATSON, N.
Dynamic turbocharged diesel engine simulator for electronic control system development.
A.S.M.E. Journal of Dynamic Systems, Measurement and Control, Vol. 106, 1984.

80. WATTS, H A, & SHAMPINE, L F.
A-Stable block implicit one step methods.
Bit, Vol 12, 1972, pp 252-266.

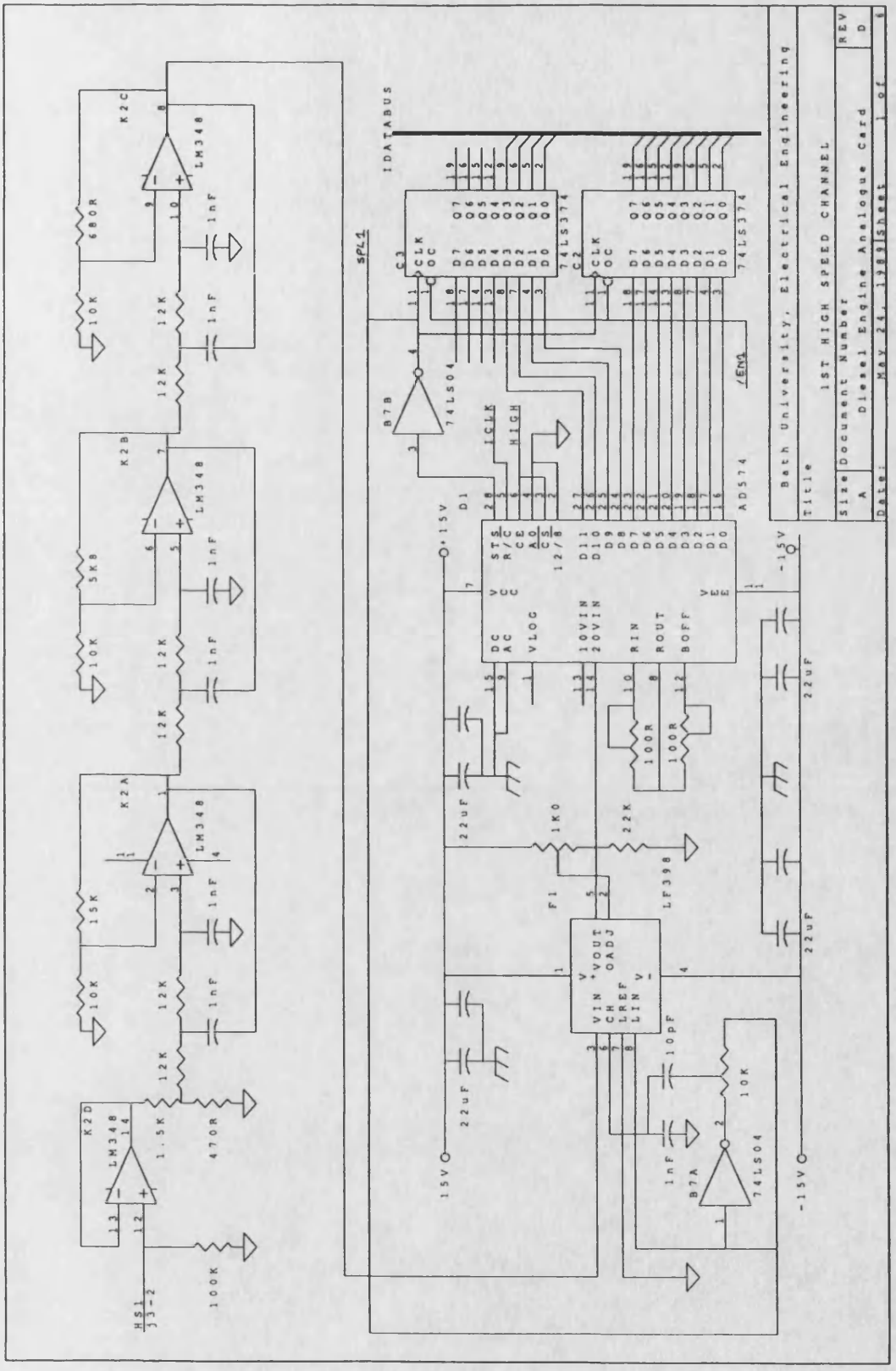
81. WEBER, H G. & BORMAN, G L.
Parametric studies using a mathematically simulated diesel engine cycle.
S.A.E. Report No. 670480, May 1967.

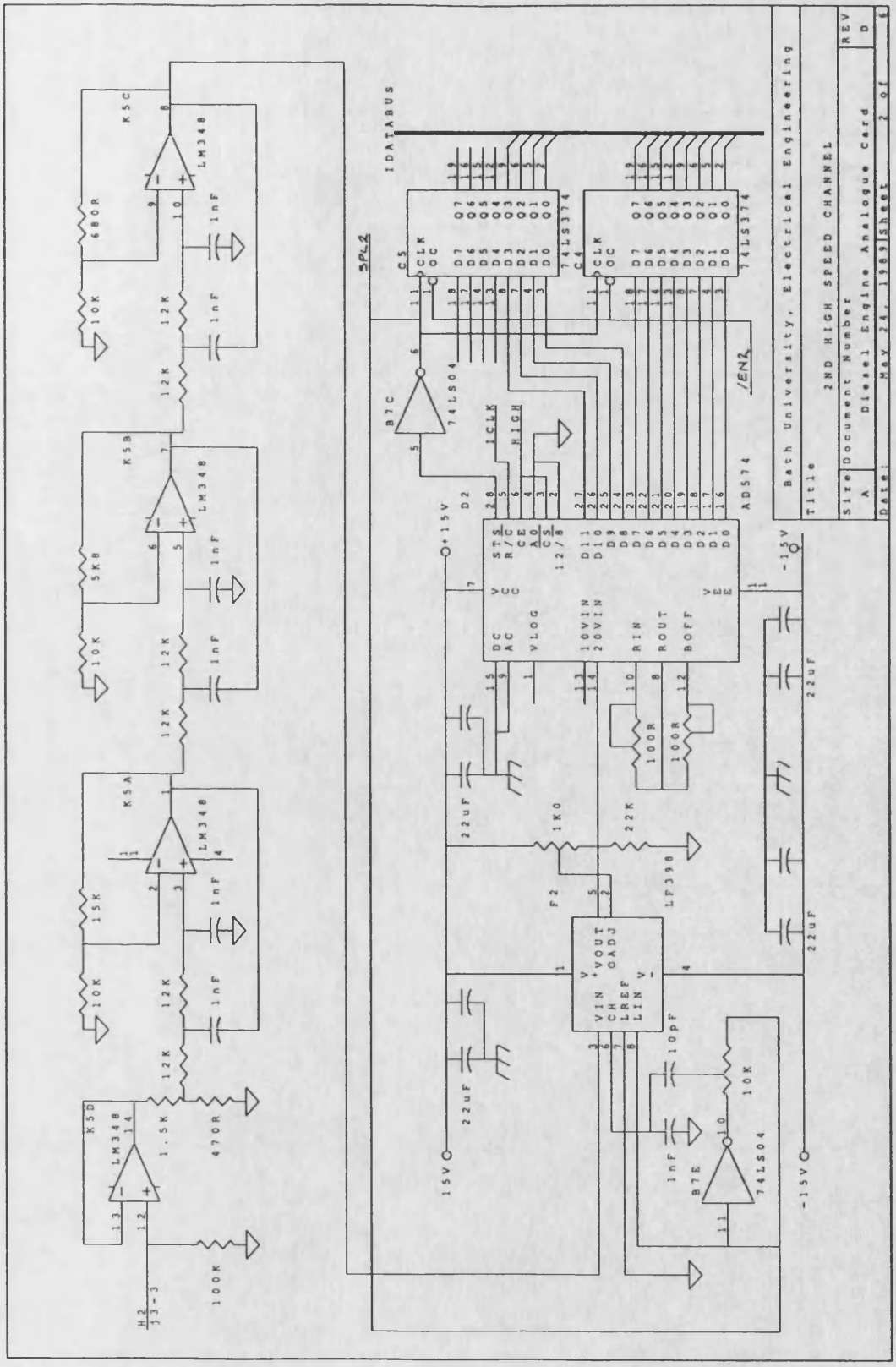
APPENDIX A.

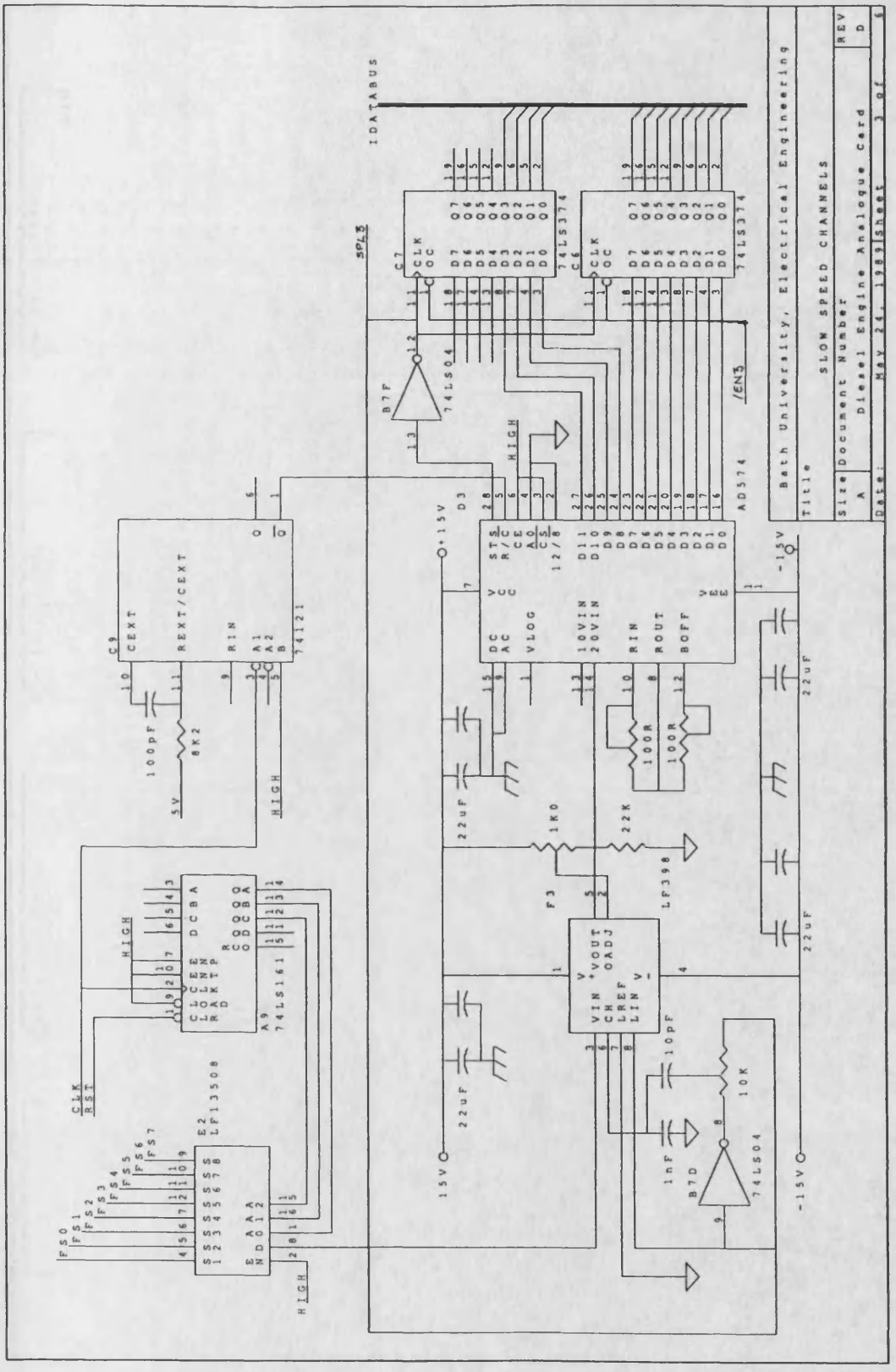
DATA ACQUISITION CARD CIRCUIT DIAGRAMS.

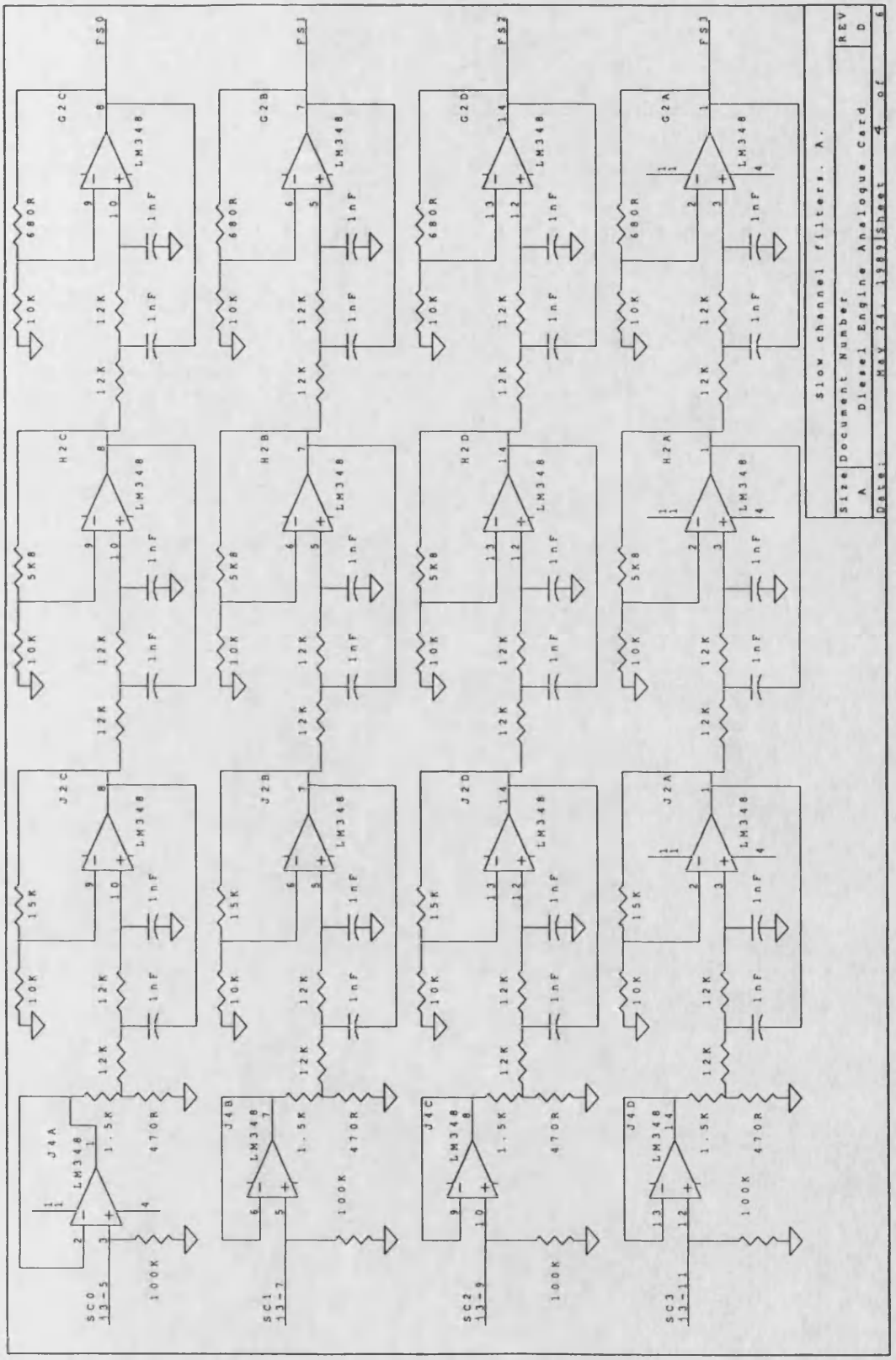
APPENDIX A.

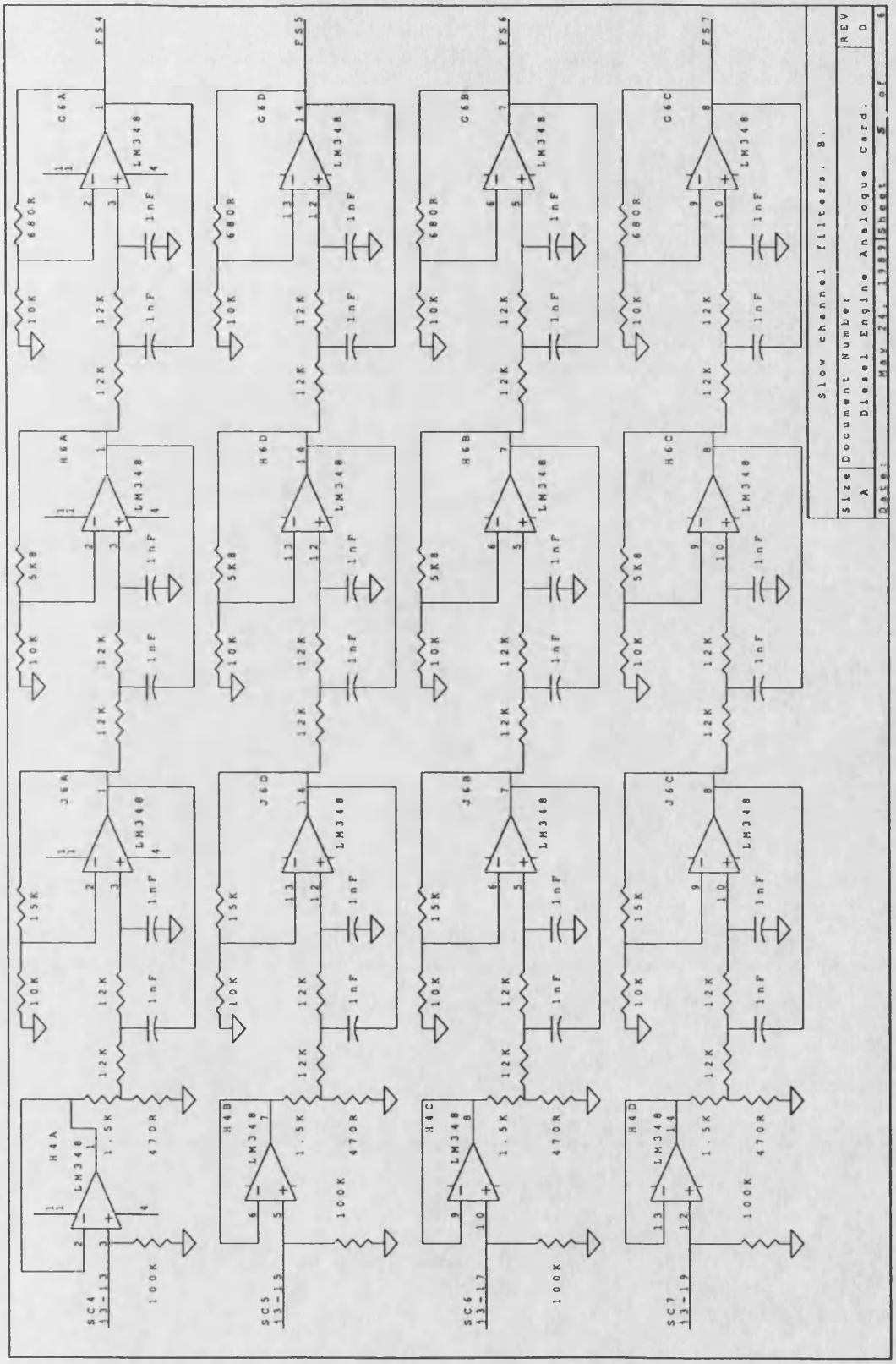
ANALOGUE DATA ACQUISITION CARD CIRCUIT DIAGRAMS.

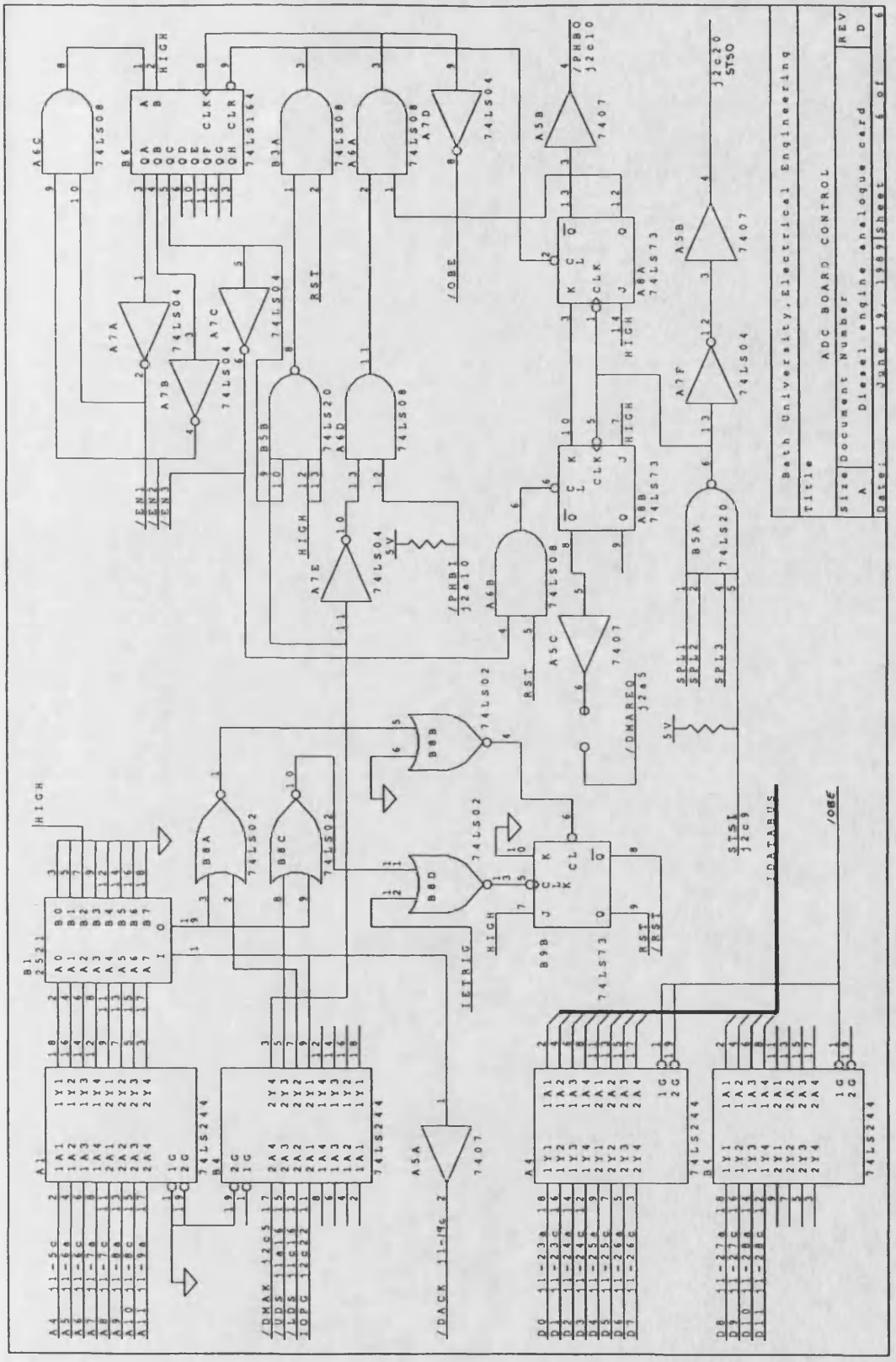

Bath University, Electrical Engineering
Title
1ST HIGH SPEED CHANNEL
Size
Document Number
Diesel Engine Analogue Card
REV
D
Date: May 24, 1988
Sheet 1 of 1
IDATABUS
SPL1
74LS374
74LS04
AD574
LM348
LF398
ICLK
HIGH
/EN4
+15V
-15V
680R
10K
12K
1nF
5K8
15K
1K0
22K
100R
22uF
10pF
1.5K
470R
100K
HS1
33-2

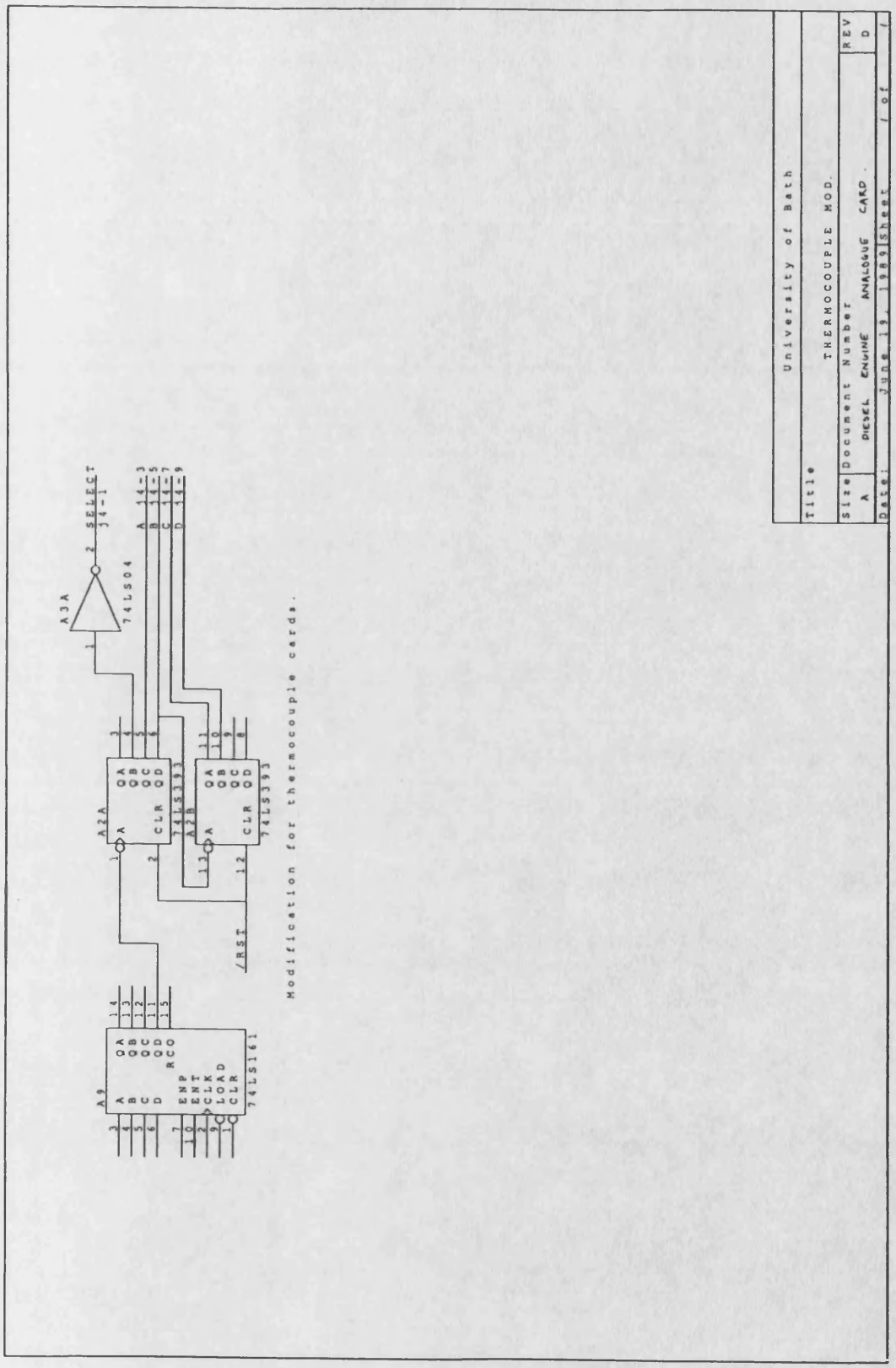

Bath University, Electrical Engineering
Title
2ND HIGH SPEED CHANNEL
Size
Document Number
REV
A
Diesel Engine Analogue Card
D
Date: MAY 24, 1989 Sheet 2 of 6
IDATABUS
SPL2
/EN2
ICLK
HIGH
AD574
74LS374
74LS04
LM348
LF398
+15V
-15V
22uF
1nF
10pF
1K0
22K
100R
680R
10K
12K
15K
5K8
1.5K
470R
100K


IDATABUS
SPL3
/EN3
HIGH
+15V
-15V
5V
Bath University, Electrical Engineering
Title
SLOW SPEED CHANNELS
Size
Document Number
REV
A
Diesel Engine Analogue Card
D
Date: May 24, 1989 Sheet 3 of 6


Slow channel filters. A.
Size A
Document Number
Diesel Engine Analogue Card
REV D
Date: May 24, 1989 Sheet 4 of 6
LM348
680R
10K
12K
1nF
5K8
15K
1.5K
470R
100K
FS0
FS1
FS2
FS3
G2C
G2B
G2D
G2A
H2C
H2B
H2D
H2A
J2C
J2B
J2D
J2A
J4A
J4B
J4C
J4D
SC0 13-5
SC1 13-7
SC2 13-9
SC3 13-11

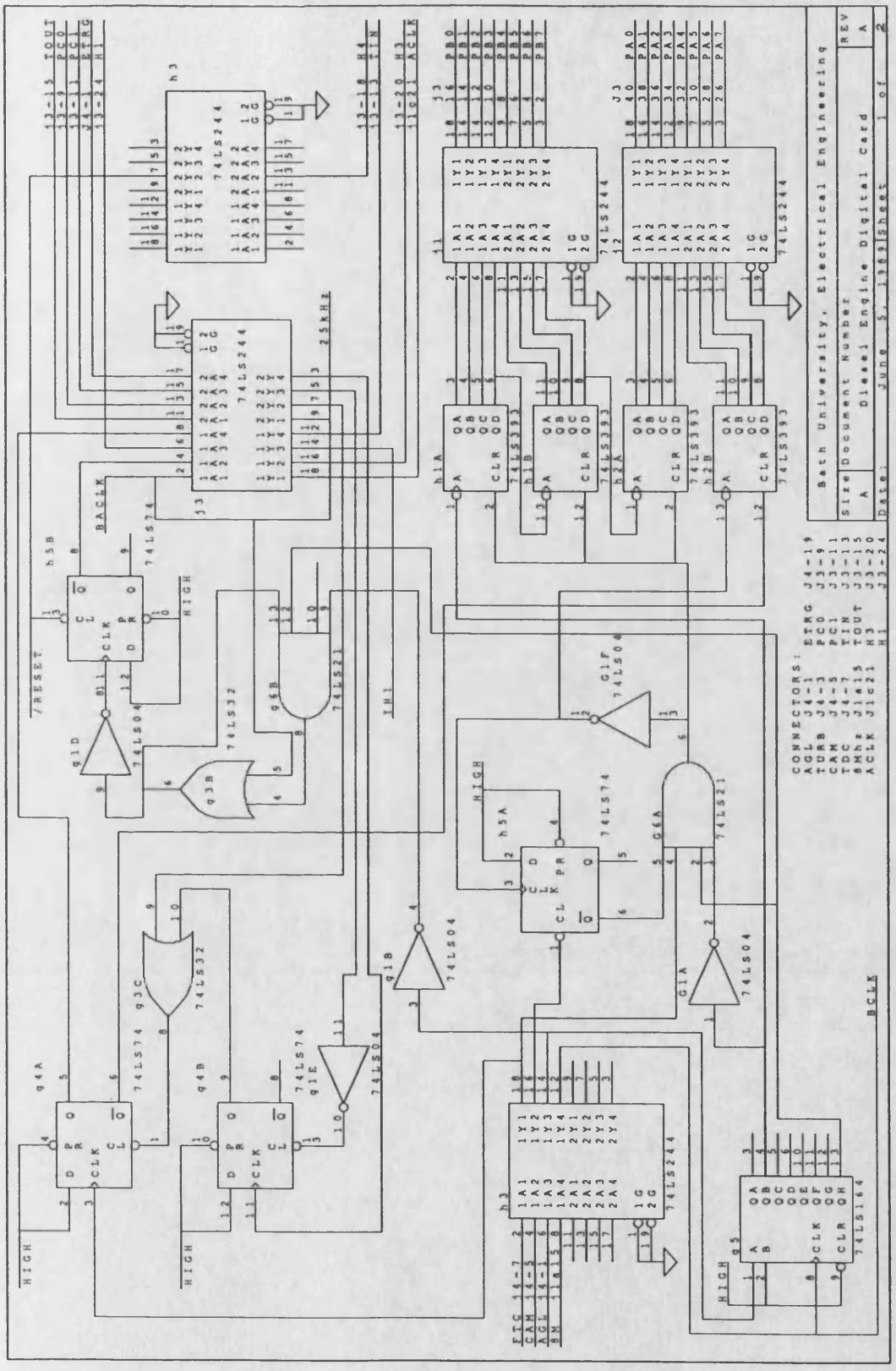

Slow channel filters. B.
Size
A
Document Number
Diesel Engine Analogue Card.
REV
D
Date: May 24, 1989 Sheet 5 of 6
FS4
FS5
FS6
FS7
G6A
G6D
G6B
G6C
H6A
H6D
H6B
H6C
J6A
J6D
J6B
J6C
H4A
H4B
H4C
H4D
LM348
680R
10K
5K8
15K
12K
1nF
1.5K
470R
100K
SC4
13-13
SC5
13-15
SC6
13-17
SC7
13-19

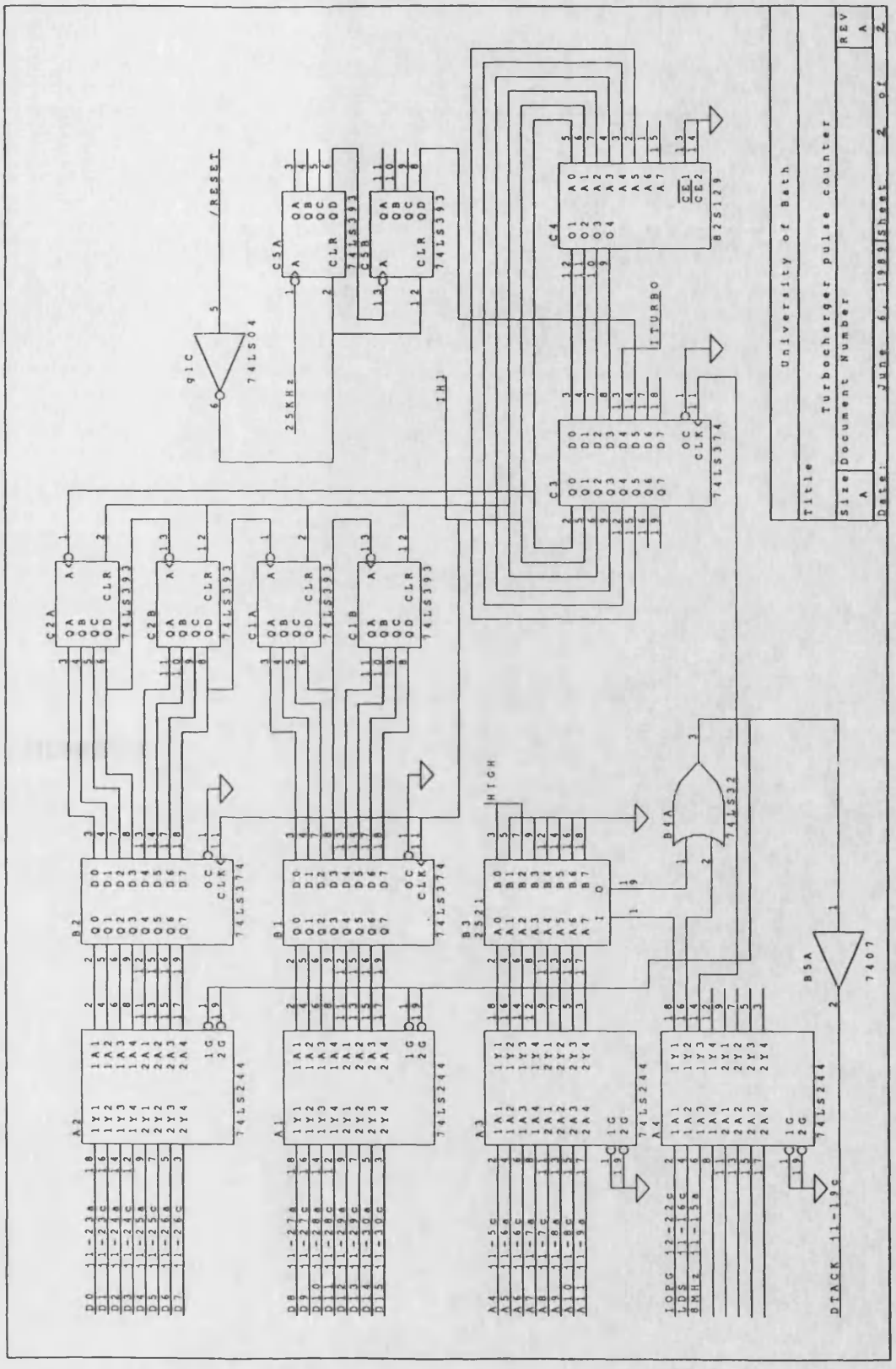

Bath University, Electrical Engineering
Title
ADC BOARD CONTROL
Size
Document Number
REV
D
A
Diesel engine analogue card
Date: June 19, 1989 Sheet 6 of 6
74LS08
74LS164
74LS04
74LS20
74LS02
74LS73
74LS244
7407
HIGH
/OBE
RST
/DMAREQ
/DMAK
/IOR
/IOW
IOPG
IETRIG
IDATABUS
STS1
STS0
/PHB0
/PHB1
/EN1
/EN2
/EN3
/DACK


University of Bath
Title
THERMOCOUPLE MOD.
Size
A
Document Number
DIESEL ENGINE ANALOGUE CARD
REV
D
Date: June 19, 1989 Sheet 1 of 1
A3A
74LS04
2 SELECT
J4-1
A 14-3
B 14-5
C 14-7
D 14-9
A2A
A2B
74LS393
/RST
A9
74LS161
ENP
ENT
CLK
LOAD
CLR
RCO
Modification for thermocouple cards.

APPENDIX A.

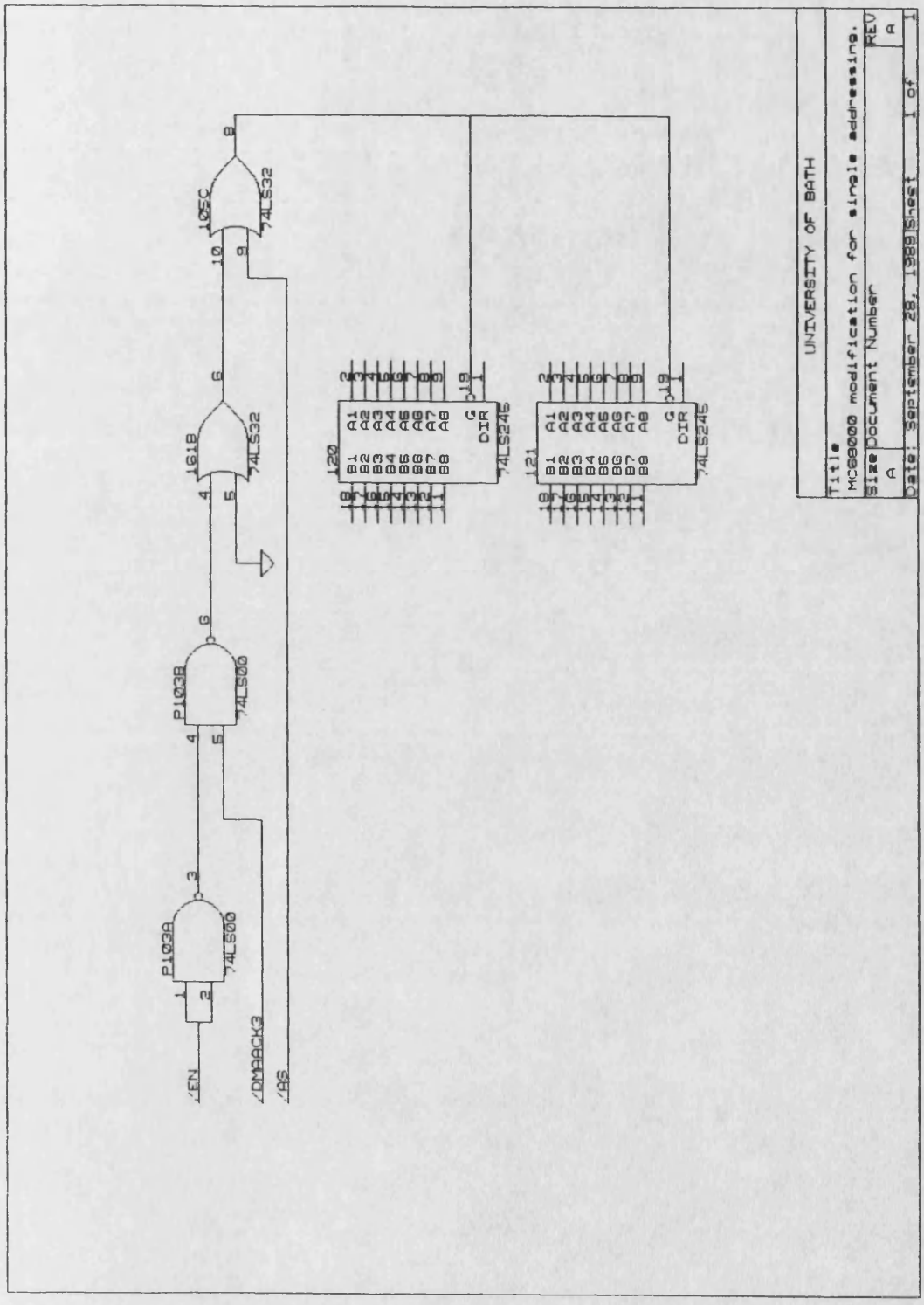
DIGITAL DATA ACQUISITION CARD CIRCUIT DIAGRAMS.


Bath University, Electrical Engineering
Diesel Engine Digital Card
CONNECTORS:
ETRG J4-19
AGL J4-1
PC0 J3-9
TDRB J4-3
PC1 J3-11
CAM J4-5
TIN J3-13
TDC J4-7
TOUT J3-15
8Mhz J1e15
H3 J3-20
ACLK J1e21
H1 J3-24
BCLK
74LS244
74LS393
74LS74
74LS04
74LS21
74LS32
74LS164
/RESET
BACLK
HIGH
25KHz


University of Bath
Title
Turbocharger pulse counter
Size
A
Document Number
REV
A
Date: June 6, 1989 Sheet 2 of 2
/RESET
25KHz
74LS04
74LS393
74LS374
74LS244
74LS32
82S129
74LS21
7407
HIGH
ITURBO
IHJ
BSA
DTACK 11-19c
IOPG 12-22c
LDS 11-16c
8MHz 11-15a

APPENDIX A.

MC68000 BOARD MODIFICATION TO USE BACK-PLANE DMA SINGLE ADDRESSING.


UNIVERSITY OF BATH
Title
MC68000 modification for single addressing.
Size
A
Document Number
REV
A
Date: September 28, 1989
Sheet 1 of 1
74LS32
74LS00
74LS245
DIR
G
/EN
/DMABACK3
/AS

APPENDIX B.

A BRIEF SPECIFICATION OF THE NEW BATH UNIVERSITY SHARED MEMORY, MULTI-TRANSPUTER COMPUTER.

The Bath University shared memory, multi-Transputer computer.

Processor chip:	Inmos T800-20 20Mhz transputer.
Computer architecture:	Shared memory
Physical structure:	Processing clusters connected via high speed fibre optic links.
Cluster structure:	16 transputer boards. Link configurer board. Optic fibre board. Overlapped cycle multi-processor bus.
Transputer boards:	1 T800 transputer. 1 Mbyte memory. Shared memory interface which allows full speed writes to all processors within a cluster. Four transputer 'link' connections.
Link configurer board:	Allows all the transputer links within a cluster to be configured into any desired network, with upto 32 IO links possible.
Fibre optic board:	This board is for inter cluster communication. The speed of communication is 125Mbaud. This gives a latency of 2μs for processor write operations between directly connected clusters.
Envisaged system:	Two clusters with a direct fibre optic link.
Ultimate system:	256 clusters arranged in a tree structure giving a maximum latency for processor write operations between clusters of 3μs

APPENDIX C.

THE LEYLAND TL11 DIESEL ENGINE SPECIFICATION.

The Leyland TL11 Diesel engine specification.

This appendix lists the physical parameters which are used in the simulation of a Leyland TL11 engine, using the 'filling and emptying' method.

Engine.	Leyland TL11 four stroke turbocharger diesel engine.
Ratings:	Maximum power 190kW at 2100rpm
No of cylinders:	6
Cylinder firing order:	1 5 3 6 2 4
Cylinder bore:	127.08 mm
Cylinder stroke:	146.05 mm
Connecting rod length:	266.7 mm
Compression ratio:	15.75 to 1
Effective inertia of engine: (including flywheel)	2.95 kgm^2
Cylinder and piston:	
Cylinder head surface area:	97.95 cm^2
Piston crown surface area:	184.44 cm^2
Cylinder bore surface area. (from top of block to 1st piston ring at TDC.)	107.01 cm^2
Combustion chamber surface area:	547.12 cm^2

Cylinder Valves.

Valve timing, relative to TDC in open period:

Exhaust valve close (EVC):	14^o
inlet valve close (IVC):	230^o
exhaust valve close (EVC):	494^o
inlet valve open (IVO):	710^o

Valve geometry.	Inlet valve.	Exhaust valve.
Head diameter:	55 mm	46.5 mm
Seat angle:	30^o	30^o
Seat width:	3.6 mm	2.8 mm
Number/cylinder:	1	1

Manifolds.

One inlet and two exhaust manifolds.

Volume of inlet manifold:	4.88 litres
Volume of front exhaust manifold:	1.38 litres
Volume of rear exhaust manifold:	1.20 litres.
Internal surface area of front exhaust manifold:	817 cm^2
Internal surface area of rear exhaust manifold:	681 cm^2

Turbocharger.

Experimental variable geometry unit (on turbine section)

Compressor:	H2C8625N Z3IU3
Turbine:	Experimental MkIIb unit.

Turbocharger inertia: 0.001kgm^2 (estimate)

Fuel System.

Direct injection system with variable injection timing.

CAV in line fuel pump type:	Majormec P5476
Standard fuel injection timing:	22o before TDC
Length of fuel delivery pipes:	0.72m

Load system.

Hydrostatic dynamometer load system, which can be set up to provide three torque speed characteristics.

- constant torque.
- constant speed.
- 'windage'

APPENDIX D.

THE MENU UTILITY ROUTINES FOR USE IN USER INTERFACES.

The menu utility routines for use in user interfaces.

menu.select (menu)

Inputs: Menu, vector of menu strings.

Action: Displays menu vector on one line in lower display box.

Result: Returns the value selected by user using return and cursor keys.

select.option1 (cursor, retabulate, menu, values)

Inputs: Menu, vector of menu strings.
Values, vector of floating point numbers.
Retabulate, TRUE/FALSE.
Cursor, TRUE/FALSE.

Action: Displays menu vector on several lines in upper display box.

Result: Returns the value selected by user using return and cursor keys.

Options:

Retabulate.	If TRUE redraws menu options.
Cursor.	If TRUE allows multiple selection of menu options, with previous choices marked.
Values.	If not zero this points to a vector of values which are displayed alongside the menu options.

page.format ()

Action: Creates basic menu title frame.

display.numbers (menu, values, retabulate, offset)

Inputs: Menu, vector of menu strings.
Values, vector of floating point numbers.
Retabulate, TRUE/FALSE
Offset, integer number.

Action: Displays menu vector on several lines in upper display box with values vector displayed alongside.

Options:

Retabulate.	If TRUE redraws menu options.
Offset.	Indicates the number of characters from left of screen at which menu is displayed.

read.a.value (string)

Action: Clears upper display area, writes string prompt and reads floating point number.

Result: Floating point number.

read.a.string (x, y, string, length)

Inputs: x,y, integer value screen coordinates.
String, vector.
Length, number of bytes in string.

Action: Clears upper display area, reads string input into string vector at point x,y on screen.